Аннотация

В представленной работе рассмотрены некоторые популярные рендеры фотореалистичных изображений, реализующие различные алгоритмы трассировки лучей. Эти алгоритмы делают возможным моделирование различных световых свойств, таких как глобальное освещение, тени, каустики, дисперсия, глубина резкости, подповерхностное рассеивание и т.д. Все алгоритмы объединяет идея симулирования пути света, но отличают реализации способов её осуществления. В зависимости от этого алгоритм может по-разному изображать различные световые эффекты, и объём вычислений будет различным. Благодаря высокой вычислительной мощности современных процессоров и видеокарт становится актуальным рассмотрение достаточно ресурсоемких алгоритмов и программ, реализующих их. В представленной работе сравниваются различные алгоритмы и их реализаций. Сделаны выводы о качестве получаемого изображения и реализованной многопоточности вычислений.

Оглавление

Определения и обозначения

Сокращение	Расшифровка
PBR	Physically-based rendering (рус. Физически-корректный рендеринг)
BSDF	Bidirectional scattering distribution function (рус. Двулучевая функция распределения рассеивания)
BRDF	Bidirectional reflectance distribution function (рус. двулучевая функция отражательной способности)
BTDF	Bidirectional transmittance distribution function (рус. Двулучевая функция пропускной способности)
BSSRDF	bidirectional scattering surface reflectance distribution function (рус. двулучевая функция распределения подповерхностного рассеивания)
PSNR	peak signal-to-noise ratio (рус. отношение пикового сигнала к шуму)
SPPM	Stochastic Progressive Photon Mapping (рус. Стохастический прогрессивный метод фотонных карт)
BDPT	Bidirectional path-tracing (рус. двунаправленная трассировка пути)

Введение

На сегодняшний день существует много программ трехмерной визуализации, которые используют для реалистичного синтеза различные алгоритмы трассировки лучей. Эти программы применяются в кино- и телеиндустрии, дизайне, архитектуре и т.п. На сегодняшний день наибольшей популярностью пользуются платные пакеты, такие как Maxwell Render, Lumion, Solidworks Visualize, Marmoset Toolbag, Corona Render, Iray, V-Ray, Indigo Render. Для независимых разработчиков и некоторых молодых компаний значительный расход средств на их приобретение может быть критическим, поэтому они склонны искать бесплатные аналоги. К счастью, такие существуют, но тогда встает вопрос о качестве их работы. Помимо цены, важными остаются такие характеристики, как время рендеринга и фотореалистичность получающегося изображения. Также важным фактором для времени работы выступает реализация распараллеливания алгоритмов рендеринга. Благодаря идее имитации пути нескольких световых лучей они легко поддерживают эту возможность. С позиций требуемой вычислительной мощности процессоров актуально рассмотреть и реализацию многопоточности у этих алгоритмов.

Таким образом, **актуальность** заключается в том, что в данной работе сравниваются эти характеристики у некоторых популярных бесплатных рендеров и исследуется их оптимальная применимость к визуализации различных световых явлений.

Объектом исследования в данной работе являются программные средства компьютерной графики, а **предметом** – алгоритмы трассировки лучей и их реализация.

Цель исследования: сравнить алгоритмы трассировки лучей, реализованные в различных бесплатных программах для трехмерной визуализации.

Задачи:

1. построить трехмерные сцены;
2. настроить идентичные параметры для каждой сцены;
3. получить эталонное изображение, с которым будет проводится сравнение;
4. запустить рендер с ограничением по времени и/или потокам;
5. построить графики скорости и качества с помощью метрики PSNR.

Глава 1 Обзор источников

Глава 1.1. Физические модели и законы

Перед тем, как говорить о реалистичном отображении внешнего мира определимся с понятием «реалистичность». Для реалистичного рендеринга алгоритмам необходимо руководствоваться законами физической оптики, т.е. корпускулярно-волновыми свойствами света, а также законами взаимодействия света с различными поверхностями.

Законы геометрической оптики

Геометрическая оптика описывает распространение электромагнитного поля как распространение лучей в пространстве. Луч – это линия (кривая или прямая), вдоль которой распространяется энергия светового поля.

1. Закон прямолинейного распространения света (принцип Ферма)

   Оптическая длина луча (произведение геометрического пути светового луча и показателя преломления среды, в которой распространяется свет) между двумя точками минимальна по сравнению со всеми другими линиями, соединяющими эти две точки. Отсюда очевидно следует, что в однородной среде световой луч движется по прямой линии.

2. Закон независимого распространения лучей

   Если через точку пространства проходит несколько лучей, то каждый луч никак не влияет на любой другой луч. То есть он ведет себя так, будто бы никаких других лучей нет.

3. Закон преломления света (Закон Снеллиуса)

   При переходе светового луча из одной среды в другую, на границе двух сред световой луч разделяется на два: отраженный и преломленный. Угол падения света на поверхность связан с углом преломления соотношением:

   $$n_1 cos\alpha = n_2 cos\beta$$

   где n_1 − показатель преломления первой среды;
   n_2 − показатель второй среды, в которую свет попадает, пройдя границу раздела;
   α− угол между падающим лучом и нормалью к поверхности в первой среде;
   β− угол между лучом и нормалью к поверхности во второй среде.

4. Законы отражения света

1) Диффузное отражение. При отражении света от шероховатой поверхности лучи отражаются в разные стороны.
2) Зеркальное отражение. Падающий и отраженный лучи, а также нормаль к отражающей поверхности в точке падения лежат в одной плоскости и угол падения равен углу отражения.
3) Полное внутреннее отражение. Ситуация, при которой интенсивность преломленного луча становится нулевой. Возникает на границе двух сред, где волна падает из среды с большим показателем преломления (т.е. более плотной).

5. Закон обратимости светового луча.

Траектория пути лучей не зависят от направления распространения. Другими словами, если луч, выпущенный из точки А в точку Б, пустить в обратном направлении от Б к А, то он будет иметь точно такую же траекторию.

Все законы геометрической оптики следуют из закона сохранения энергии. Также они не являются независимыми друг от друга. [2]

Электромагнитные свойства света

Свет (или световое поле) – это электромагнитные волны в оптическом диапазоне длин волн λ (0 – 40 мкм). В этом диапазоне выполняются законы геометрической оптики и свет слабо взаимодействует с веществом. Оптический диапазон состоит из 4 видов излучения: рентгеновское, ультрафиолетовое, инфракрасное и видимое. В данной работе имеет значение только последнее.

Видимое излучение – отраженный от разных поверхностей свет видимого диапазона, лежащий в пределах от 380 нм до 780 нм [1].

Дифракция

Под дифракцией света понимают всякое уклонение от прямолинейного распространения света, если оно не может быть истолковано как результат отражения, преломления или изгибания световых лучей в средах с непрерывно меняющимся показателем преломления. Если в среде имеются мельчайшие частицы постороннего вещества (туман) или показатель преломления заметно меняется на расстояниях порядка длины волны, то в этих случаях говорят о рассеянии света[1].

Интерференция и когерентность

Колебания (и волны) называются когерентными, если разность их фаз δ постоянна во времени[1]. Если оба колебания не согласованы друг с другом т.е. разность их фаз δ как-то изменяется во времени, то такие колебания называют некогерентными. При суперпозиции когерентных волн происходит перераспределение интенсивности I в пространстве: в одних местах возникают максимумы, в других минимумы интенсивности. Такое явление называется интерференцией волн.

Принцип Гюйгенса-Френеля

Окружим все источники света S1, S2, S3, ... произвольной замкнутой поверхностью F. Каждую точку такой поверхности можно рассматривать как источник вторичных волн, распространяющихся во всех направлениях. Эти волны когерентны, поскольку все они возбуждаются одними и теми же первичными источниками. Световое поле, возникающее в результате их интерференции, в пространстве вне поверхности F совпадает с полем реальных источников света. [1]

Таким образом, действительные источники света можно как бы заменить окружающей их светящейся поверхностью F с непрерывно распределенными по ней когерентными вторичными источниками. Отличие этой поверхности от реальной поверхности излучающего тела состоит в том, что она абсолютно прозрачна для всякого излучения. В такой формулировке принцип Гюйгенса-Френеля выражает весьма общее положение. Он означает, что волна, отделившаяся от своих источников, в дальнейшем ведет автономное существование, совершенно не зависящее от наличия источников.

Каустики

Каустика – это огибающая световых лучей, преломленных или отраженных от некоторой кривой поверхности или объекта, или это проекция этой огибающей на другую поверхность. Каустика возникает из-за концентрации световых лучей в определенной точке. [1]

Закон Ламберта

Закон Ламберта утверждает, что интенсивность излучения, отражаемая от идеальной диффузной поверхности (или излучаемая идеальным диффузным источником), прямо пропорциональна косинусу угла θ между направлением падающего света и нормалью к поверхности. [2]

Уравнения Френеля

Уравнения Френеля описывают зависимость амплитуды электрического (магнитного) поля падающей волны и полей отраженной и преломленной волн [1]:

$$r_\perp \equiv \frac{R_\perp}{\mathcal{E}_\perp} = \frac{n_1 \cos\varphi - n_2 \cos\psi}{n_1 \cos\varphi + n_2 \cos\psi}, \quad d_\perp \equiv \frac{D_\perp}{\mathcal{E}_\perp} = \frac{2n_1 \cos\varphi}{n_1 \cos\varphi + n_2 \cos\psi},$$

$$r_{||} \equiv \frac{R_{||}}{\mathcal{E}_{||}} = \frac{n_2 \cos\varphi - n_1 \cos\psi}{n_2 \cos\varphi + n_1 \cos\psi}, \quad d_{||} \equiv \frac{D_{||}}{\mathcal{E}_{||}} = \frac{2n_1 \cos\varphi}{n_2 \cos\varphi + n_1 \cos\psi}.$$

Здесь $r_\perp -$ составляющая отраженной волны, лежащая перпендикулярно плоскости падения;

$d_\perp -$ составляющая преломленной волны, лежащая перпендикулярно плоскости падения;

$r_{||} -$ составляющей отраженной волны, лежащая в плоскости падения;

$d_{||} -$ составляющей преломленной волны, лежащая в плоскости падения;

$R_\perp -$ амплитуда составляющей отраженной волны, лежащая перпендикулярно плоскости падения;

$R_{||} -$ амплитуда составляющей отраженной волны, лежащая в плоскости падения;

$D_\perp -$ амплитуда составляющей преломленной волны, лежащая перпендикулярно плоскости падения;

$D_{||} -$ амплитуда составляющей преломленной волны, лежащая в плоскости падения амплитуда составляющей отраженной волны, лежащая перпендикулярно плоскости падения;

$\mathcal{E}_\perp -$ амплитуда составляющей падающей волны, лежащая перпендикулярно плоскости падения;

$\mathcal{E}_{||} -$ амплитуда составляющей падающей волны, лежащая в плоскости падения

$\varphi -$ угол падения, $\psi -$ угол преломления;

$n_1, n_2 -$ показатели преломления 1-й и 2-й сред.

Эти отношения называются коэффициентами Френеля. При обыкновенном отражении угол ψ, а с ним и все коэффициенты Френеля вещественны. При скользящем падении, когда угол φ близок к 90°, отражение практически полное.

$$\frac{R_\perp}{\mathcal{E}_\perp} = \frac{R_\parallel}{\mathcal{E}_\parallel} \to -1$$

С этим связана видимость подводных объектов вбили к наблюдателю и постепенным «размытием» при отдалении. Этим же объясняется, почему изображение заходящего солнца в тех же условиях по яркости почти не уступает самому солнцу.

Подповерхностное поглощение и рассеяние света

При прохождении электромагнитной волны света через какое-либо вещество энергия этой волны расходуется на возбуждение колебаний электронов. Часть этой энергии переходит в другие виды энергии, а часть возвращается в виде вторичных волн, вызванных колебанием. Тут следует учесть взаимную интерференцию таких волн: в однородной среде вторичные волны полностью гасят друг друга, поэтому рассеяние света возникает только в неоднородной среде. Дифракционная картина в такой среде выглядит как равномерное распределение интенсивности света во всех направлениях и называется рассеянием света. Поглощением же света называется явление уменьшения энергии и, вследствие, интенсивности, световой волны при ее распространении в веществе из-за преобразования энергий. [2]

В настоящее время не существует способа полностью сымитировать сложное поведение света, учитывая все его физические свойства. Однако в 1986 году на конференции SIGGRAPH Джеймс Т. Каджия [4] представил интегральное уравнение, которое является основой для имитирования любых физически-корректных материалов. Подробно о нем далее.

Глава 1.2 Уравнение рендеринга и PBR.

Энергетическая яркость

Энергетическая яркость – радиометрическая величина, которая характеризует наблюдаемую энергию, другими словами, количество света, на некоторой площади. Чтобы дать точное определение, необходимо предварительно разъяснить следующие понятия:

1. Телесный угол ω – это объемный угол, который характеризует долю пространства, которая может быть спроецирована на некоторую поверхность, обычно сферу, площади S (рис. 1). Мерой такого угла является отношение площади S к квадрату радиуса сферы r : $\omega = \frac{S}{r^2}$. В системе СИ поверхность с площадью r^2 вырезанная из сферы радиуса r является телесным углом в 1 Стерадиан.

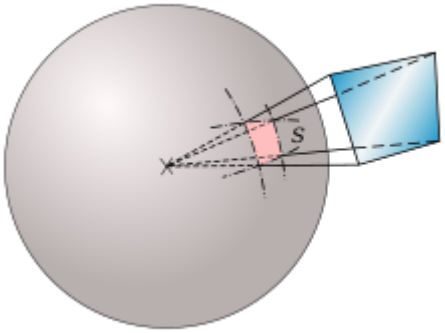


Рис. 1 Телесный угол

2. Поток излучения Ф – мощность излучения, определяемая отношением энергии, переносимой излучением, ко времени переноса, значительно превышающему период электромагнитных колебаний. Энергия, испускаемая источником света, представляется как функция, зависящая от длин волн. Однако использование самих длин волн в компьютерной графике нецелесообразно, поэтому используются упрощения, такие как RGB.
3. Сила излучения I – физическая величина, определяемая отношением потока излучения Ф, распространяющегося от источника излучения внутри малого телесного угла ω, содержащего рассматриваемое направление, к этому углу: $I = \frac{d\Phi}{d\omega}$

Энергетическая яркость (*или лучистость*) – физическая величина, определяемая отношением потока излучения Ф к произведению телесного угла ω, в котором он распространяется, и проекции площади A, излучающего элемента поверхности на плоскость, перпендикулярную нормали к поверхности: $L = \frac{d^2\Phi}{dAd\omega}$

Уравнение рендеринга

Уравнение рендеринга вычисляет энергетическую яркость точки поверхности x, исходящую в направлении ω_0 (направление условного наблюдателя), как сумму собственной энергетической яркости поверхности L_e и отраженного излучения, которое выражается интегралом. В частности, когда поверхность не излучает свет, т.е. при $L_e = 0$, выходит *уравнение отражения*.

$$L_0(x, \overrightarrow{\omega_0}) = L_e(x, \overrightarrow{\omega_0}) + \int_{\Omega} \left(f_r(x, \overrightarrow{\omega_0}, \overrightarrow{\omega_i}) \cdot L_i(x, \overrightarrow{\omega_i}) \cdot (\overrightarrow{\omega_i}, \vec{n})\right) d\omega_i$$

Поверхность S телесного угла принимается бесконечно малой и фактически он становится вектором направления $\vec{\omega}$, а поверхность A – точкой x.

В интеграле:

1) $\int_{\Omega}(\dots) d\overrightarrow{\omega_i}$ – интеграл по полусфере $\Omega = S^2$, необходимый для учета всех входящих векторов $\overrightarrow{\omega_i}$,
2) $L_i(x, \overrightarrow{\omega_i})$– эн. яркость падающего света в точку x, направленная вдоль вектора $\overrightarrow{\omega_i}$,
3) $(\overrightarrow{\omega_i}, \vec{n}) = \cos\theta_i$ – скалярное произведение векторов, которое обозначает угол между входящим (падающим) вектором $\overrightarrow{\omega_i}$ и вектором нормали к поверхности $\vec{n}$,
4) $f_r(x, \overrightarrow{\omega_0}, \overrightarrow{\omega_i}) = \frac{dL_0}{L_i \cdot \cos\theta_i d\omega_i}$ – двулучевая функция отражательной способности (англ. Bidirectional reflectance distribution function [BRDF]).
5) Также существует BTDF (bidirectional transmittance distribution function) – Двулучевая функция пропускной способности, которая определяет количество поглощенного света и BSDF (Bidirectional Scattering Distribution Function) – Двулучевая функция распределения рассеивания, которая объединяет BRDF и BTDF и которая также, в зависимости от реализации, подставляется в уравнение рендеринга. Более конкретно об этом далее.

Аналитического решения этого уравнения нет.

Уравнение рендеринга не учитывает некоторых физических явлений света, например:

- Коэффициент прохождения волн;
- Подповерхностное рассеивание;
- Поляризацию;
- Интерференция.

Но это практически не влияет на качество получаемых изображений.

PBR или Физически-корректный рендеринг

Physically-Based Rendering – концепция, которая основываясь на физических законах, стремиться добиться максимальной реалистичности в симулировании материалов и, тем самым, фотореалистичности получаемых рендерингом изображений. Физически-корректный материал такие вещи, как:

1) Теория микрограней – поверхность имеет микроскопические неровности.
2) Закон сохранения энергии – поверхность не может отразить больше энергии, чем принимает
3) Закон Френеля – поверхности имеют разную отражательную способность в зависимости от угла просмотра.
4) Альбедо – учет отражения материалом волн определенной длины, другими словами, чистый цвет
5) Проводник/диэлектрик – учет способности проводников и диэлектриков по-разному отражать свет, (60-90%) и (0-20%) соответственно.

Теория микрограней (Microfacet Theory)

В основе этой теории лежит представление о поверхности сред, как поверхности с микроскопическими неровностями, но, тем не менее, достаточно большими, по сравнению с длиной волны. [3] При падении световых лучей на такую поверхность может возникнуть три ситуации:

1. Самозатенение (англ. self shadowing) – луч света не долетает до поверхности, от которой он должен отразиться
2. Самоперекрытие (англ. masking) – отраженный луч сталкивается с препятствием и не вылетает.
3. Многократное отражение (англ. retroreflection) – ситуация, когда луч отражается от поверхности несколько раз, перед тем как улететь.

Таким образом, если микрограни ориентированы в разные стороны, то лучи рассеиваются в разные стороны – поверхность более шероховатая. В компьютерной графике используется коэффициент шероховатости, число принадлежащее отрезку [0, 1], применяемое к медианному вектору h. Он определяет количество микрограней, направленных вдоль него: чем их меньше, тем более шероховатая поверхность.

Вектор $h = \frac{\omega_0+\omega_i}{||\omega_0+\omega_i||}$, где

- ω_0 – вектор, направленный в сторону наблюдателя,
- ω_i – вектор, направленный вдоль падающего света

Закон сохранения энергии

Как уже упоминалось ранее, при столкновении с поверхностью часть света отражается, а часть поглощается. Из этого очевидно следует простой вывод: свет,

который отражается от поверхности никак не может быть ярче света, который на нее падает.

Это реализуется за счет аналогичного разделения на зеркальную, которая отвечает за отражение, и диффузную, которая соответствует поглощению, компоненты. Тут следует уточнить, что концепция PRB не учитывает вторичные волны, т.е. подповерхностное рассеивание, которое реализуется другими способами, например, с помощью функции BSSRDF (Bidirectional Surface Scattering Distribution Function) - Двунаправленная Функция Поверхностного Рассеивания и Распределения [5].

Альбедо

Альбедо (англ. Albedo) – характеристика диффузной отражательной способности поверхности. Это отношение отражённого (рассеянного) потока излучения к потоку падающего излучения. В компьютерной графике реализуется с помощью *карт альбедо* – текстур, содержащих только информацию о цвете поверхности в виде тройки RGB для каждого пикселя. [6]

Металлы

Геометрические законы отражения света от металлов такие же, что и для непоглощающих сред. Различие есть лишь в законах преломления. Преломленные волны затухают очень быстро, что по сути означает, что у металлов нет вторичных волн, а, следовательно, рассеивания. Таким образом у металлических поверхностей нет диффузной компоненты, а есть только зеркальная.

Ядром Физически-Корректного рендеринга является уравнение рендеринга и функция BSDF. Последняя определяет, как свет взаимодействует с веществом. Существует большое количество разных вариантов этой функции и от неё и ее реализации сильно зависит реалистичность поверхности. Другими словами, она является определяющей материала. В данной работе подробно рассмотрены только некоторые из этих вариантов, в контексте исследуемых рендеров и алгоритмов.

Двулучевая функция распределения рассеивания (BSDF)

BRDF – функция, которая на основе свойств поверхности, определяет количество отраженного света, другими словами, вклад, который вносит каждый падающий луч ω_i. Она является характеристикой конкретной поверхности, возвращает долю энергии, отраженной от материала в зависимости от направления ω_0.

Для того чтобы мы могли считать заданную BRDF физически корректной, она должна удовлетворять двум базовым требованиям – принципу взаимности Гельмгольца (Helmholtz reciprocity) и закону сохранения энергии [20]:

1. Принцип взаимности Гельмгольца говорит о том, что если мы обратим направление движения света (т. е. поменяем местами источник и приемник света), то ничего не изменится, другими словами, BRDF симметрична:
$$f_r(\overrightarrow{\omega_0}, \overrightarrow{\omega_i}) = f_r(\overrightarrow{\omega_i}, \overrightarrow{\omega_0})$$
2. Поверхность не может отражать больше световой энергии, чем на нее падает. Вся падающая световая энергия может быть выражена в виде интеграла по верхней полусфере S^2:
$$\int_{S^2} f_r(x, \overrightarrow{\omega_0}, \overrightarrow{\omega_i}) \cdot \cos\theta_i \cdot d\omega_i$$

В рендерах BSDF реализуется посредством шейдеров – самостоятельных программ, которые принимают множество параметров, определяющих оптические свойства поверхности. Рассмотрим виды, особенности и реализации данной функции, используемые в исследовании.

В рендере **appleseed**:

1. Physically-based glass BSDF

Данная функция используется в шейдере asGlass. Она реализована, основываясь на статье [3]. Все нижеследующие обозначения взяты оттуда.

$f_s -$ BSDF, $f_r -$ BRDF, $f_t -$ BTDF.

$f_s(\mathbf{o}, \mathbf{i}, \mathbf{m}) = f_r(\mathbf{o}, \mathbf{i}, \mathbf{m}) + f_t(\mathbf{o}, \mathbf{i}, \mathbf{m})$, где

$\mathbf{o} -$ вектор, направленный в сторону отражения;
$\mathbf{i} -$ вектор направления падающего луча;
$\mathbf{m} -$ вектор микронормали, т.е. нормали к конкретной микрограни.
f_r регулирует отраженный свет и имеет вид:
$$f_r(\mathbf{o}, \mathbf{i}, \mathbf{n}) = \frac{\mathrm{F}(\mathbf{i}, \mathbf{h_r})\mathrm{G}(\mathbf{i}, \mathbf{o}, \mathbf{h_r})\mathrm{D}(\mathbf{h_r})}{4|\mathbf{i} \cdot \mathbf{n}||\mathbf{o} \cdot \mathbf{n}|}$$

$\mathbf{n}$ –усредненный, по сути, обычный вектор нормали к поверхности;

$\mathbf{h_r} = \mathbf{h_r}(\mathbf{i}, \mathbf{o}) = \frac{\overrightarrow{\mathrm{h_r}}}{\|\mathrm{h_r}\|}$, где $\overrightarrow{\mathrm{h_r}} = \mathrm{sign}(\mathbf{i} \cdot \mathbf{n})(\mathbf{i} + \mathbf{o})$ – нормализованный вектор;

$\overrightarrow{\mathrm{h_r}}$ – вектор направления отраженного луча.

Данная модель функции называется BRDF Кука-Торренса. [17]

f_t регулирует преломленные лучи и имеет вид:
$$f_t(\mathbf{o}, \mathbf{i}, \mathbf{n}) = \frac{|\mathbf{i} \cdot \mathbf{h_t}||\mathbf{o} \cdot \mathbf{h_t}|}{|\mathbf{i} \cdot \mathbf{n}||\mathbf{o} \cdot \mathbf{n}|} \frac{\eta_o^2(1 - \mathrm{F}(\mathbf{i}, \mathbf{h_t}))\mathrm{G}(\mathbf{i}, \mathbf{o}, \mathbf{h_t})\mathrm{D}(\mathbf{h_t})}{(\eta_\mathrm{i}(\mathbf{i} \cdot \mathbf{h_t}) + \eta_\mathrm{o}(\mathbf{o} \cdot \mathbf{h_t}))}$$

η_o – Показатель преломления второй среды, в которую преломился луч;

η_i – Показатель преломления первой среды, в которую луч отразился;

$\mathbf{h_t} = \mathbf{h_t}(\mathbf{i}, \mathbf{o}) = \frac{\overrightarrow{h_t}}{\|h_t\|}$, где $\overrightarrow{h_t} = -(\eta_i \mathbf{i} + \eta_o \mathbf{o})$ – нормализованный вектор;

$\overrightarrow{h_t}$ – вектор направления преломленного луча.

$F(i, h) -$ функция уравнения Френеля (Fresnel equation).
Уравнения Френеля описывают отношение отраженного и преломленного света, которое зависит от угла, под который мы смотрим на поверхность. Так как они слишком сложны для вычислений, функция F является аппроксимацией. Она возвращает процент отраженного света на основании угла, под которым камера видит эту поверхность. Следующее выражение является удобной точной формулировкой для диэлектриков и неполяризованного света:

$$F(\mathbf{i}, \mathbf{m}) = \frac{(g-c)^2}{2(g+c)^2}\left(1 + \frac{(c(g+c)-1)^2}{(c(g-c)+1)^2}\right),$$

где $g = \sqrt{\frac{\eta_t^2}{\eta_i^2} - 1 + c^2}$; $c = |\mathbf{i} \cdot \mathbf{m}|$;
η_t – Показатель преломления второй среды, в которую преломился луч;
η_i – Показатель преломления первой среды.

$G(i, o, h) -$ функция геометрии (Geometry function), которая статистически аппроксимирует относительную площадь поверхности, где происходит самозатенение и перекрытие. Эта функция зависит от функции нормального распределения D и деталей микроповерхности. В общем случае используется приближение Смита [10], которое аппроксимирует двунаправленное самозатенение и перекрытие с помощью двух однонаправленных коэффициентов, возвращаемых G_1 :

$$G(\mathbf{i}, \mathbf{o}, \mathbf{m}) \approx G_1(\mathbf{i}, \mathbf{m}) G_1(\mathbf{o}, \mathbf{m})$$

где G_1 зависит от функции нормального распределения D

$D(h) -$ функция нормального распределения (Normal Distributions Functions). Она аппроксимирует количество микрограней поверхности, ориентированных по медианному вектору, основываясь на шероховатости поверхности (это основная функция, аппроксимирующая микрограни). Существует несколько вариантов функций распределения, которые дают немного разный результат. В шейдере можно выбрать какой-нибудь из них.

1) Распределение Бэкманна (Beckmann):

$$D(\mathbf{m}) = \frac{\chi^+(\mathbf{m}\cdot\mathbf{n})}{\pi\alpha_b^2\cos^4\theta_m}\, e^{\frac{-\tan^2\theta_m}{\alpha_b^2}},$$

где $\chi^+(a) = \begin{cases}1, a > 0\\ 0, a < 0\end{cases}$; (т.е. D(**m**)=0 если $\mathbf{m}\cdot\mathbf{n} \le 0$)

$\theta_m = \arctan\sqrt{-\alpha_b^2\log(1-\xi_1)}$ – угол между **m** и **n**;

ξ_1 – равномерно распределенная случайная величина на интервале (0, 1]

α_b – параметр шероховатости для распр. Бэкманна;

Соответствующая D функция G_1:

$$G_1(\mathbf{v},\mathbf{m}) = \chi^+\left(\frac{\mathbf{v}\cdot\mathbf{m}}{\mathbf{v}\cdot\mathbf{n}}\right)\frac{2}{1+\mathrm{erf}(a)+\frac{1}{a\sqrt{\pi}}\,e^{-a^2}},$$

где $a = (\alpha_b \tan\theta_v)^{-1}$;

$\mathrm{erf(x)} = \frac{2}{\sqrt{\pi}}\int_0^x e^{-x^2}dx$ – функция погрешности;

$\chi^+ -$ выполняет условие однонаправленность (т.е. вектор **v** находится на одной стороне микро- и макроповерхности)

Существует аппроксимация, которая приводится в статье, с погрешностью менее 0.35 %:

$$G_1(\mathbf{v},\mathbf{m}) = \chi^+\left(\frac{\mathbf{v}\cdot\mathbf{m}}{\mathbf{v}\cdot\mathbf{n}}\right)\begin{cases}\dfrac{3.535a+2.181a^2}{1+2.276a+2.577a^2}, & a < 1.6\\ 1, & \text{в другом случае}\end{cases}$$

2) Распределение Фонга (Phong):

$$D(\mathbf{m}) = \chi^+(\mathbf{m}\cdot\mathbf{n})\frac{\alpha_p+2}{2\pi}(cos\theta_m)^{\alpha_p}$$

α_p - параметр шероховатости для распр. Фонга;

Функция $G_1(\mathbf{v},\mathbf{m})$ такая же, ка и в распределении Бэкманна, за исключением того, что $a = \frac{\sqrt{0.5\alpha_p+1}}{\tan\theta_v}$; $\theta_m = \arccos(\xi_1^{\frac{1}{\alpha_p+2}})$.

3) Распределение GGX

$$D(\mathbf{m}) = \frac{\alpha_g^2\,\chi^+(\mathbf{m}\cdot\mathbf{n})}{\pi cos^4\theta_m\left(\alpha_g^2+\tan^2\theta_m\right)^2},$$

где α_g – параметр шероховатости для распр. GGX;

$$G_1(\mathbf{v}, \mathbf{m}) = \chi^+\left(\frac{\mathbf{v}\cdot\mathbf{m}}{\mathbf{v}\cdot\mathbf{n}}\right)\frac{2}{1+\sqrt{1+\alpha_g^2\tan^2\theta_v}}$$

$$\theta_m = \arctan\left(\frac{\alpha_g\sqrt{\xi_1}}{\sqrt{1-\xi_1}}\right)$$

Замечание. *Все описываемые далее материалы поверхностей имеют похожую на вышеописанную структуру: в них так же используются функции $f_s - BSDF$, $f_r - BRDF$, $f_t - BTDF$, а также функции D, F и G и различается только их реализация. Поэтому следующие модели будут описаны коротко с ссылками на литературу.*

2. Disney's principled, layered BRDF [32]

Данный материал основан на разработке компании Disney, которая искала оптимальный способ физически-корректно симулировать практически любую поверхность, сохраняя простоту в использовании.

Principled BRDF основан на 5 принципах:

1) Следует использовать интуитивные понятные, а не физические параметры.
2) Должно использоваться как можно меньше параметров.
3) Параметры должны варьироваться в диапазоне от нуля до единицы.
4) Параметры должны позволять выходить из диапазона, обеспечивающего реалистичность, там, где это имеет смысл.
5) Любые комбинации параметров должны также давать максимально реалистичный результат. Сложные материалы – это смеси более простых, например, диффузных, диэлектрических, металлических, прозрачных покрытий и т. д. С помощью параметров, которые регулирует пользователь, можно получить большое количество уникальных материалов.

В качестве диффузной модели используется [20]:

$$f_r = \frac{c}{\pi}(1+(F_{D90}-1)(1-\cos\theta_r)^5)(1+(F_{D90}-1)(1-\cos\theta_i)^5)$$

где c – альбедо;

θ_r - угол между нормалью n и вектором r, направленным в камеру;

θ_i – угол между нормалью n и вектором i, направленным на источник света;

$F_{D90} = \frac{1}{2} + 2\,(\cos\theta_d)^2 \cdot \alpha$, где

θ_d – это угол между векторам i и нормалью микрограни h;

α – коэффициент шероховатости.

Для этой модели освещения не выполнен закон сохранения энергии, но это было сознательным решением, направленным на удобство художников.

В число его параметров входит:

1) Surface Color – цвет поверхности, обычно поставляемый текстурными картами;
2) Subsurface Amount – управляет диффузной формой с использованием подповерхностного приближения;
3) Specular Amount – степень гладкости поверхности;
4) Specular Roughness – кажущаяся шероховатость поверхности, влияет на зеркальные блики;
5) Specular Tint – количество оттенков зеркальных бликов. Значение 0 соотв. существованию ахроматических бликов, и значение 1.0 – сохраняет цвет поверхности.
6) Metallicness – металлическое состояние (0 – диэлектрик, 1 – металл). Это линейная смесь между двумя различными моделями. Металлическая поверхность, как об этом говорилось ранее, не имеет диффузного компонента, так как весь преломленный свет затухает. Остается только зеркально-отраженный свет.
7) Anisotropy Amount – величина анизотропности поверхности, влияющая на зеркальные блики: при значении 0.0 создает изотропные блики, а при значении 1.0 – полностью анизотропные блики.
8) Anisotropy Angle – угол поворота анизотропных бликов. Отображает значения из диапазона [0,1] в диапазон [0,360].

Здесь описаны только некоторые из них, задействованные в материалах сравниваемых моделей. Данная функция реализована в шейдере asDisneyMaterial. Она основана на курсе лекций и соответствующих статьях, представленных на конференции ACM SIGGRAPH 2012. [11]

3. Physically-based metal BRDF [12]

Свет, отражающийся от металлических поверхностей, описывается уравнениями Френеля, которые управляются комплексным показателем преломления $\eta = n+ik$. Вместе n и k определяют две характеристики кривой Френеля для материала: отражательную способность материала при прямом просмотре поверхности, когда нормаль направлена в камеру и скорость перехода к зеркальному отражению света в зависимости от угла просмотра. Однако, поскольку обе этих характеристики зависят

от обоих параметров *n* и *k*, это затрудняет настройку материала. В статье [12] описан и реализован метод переназначения аппроксимированных уравнений Френеля для неполяризованных волн с параметрами *n* и *k* на более интуитивные параметры *r* и g, которые характеризуют цвет поверхности прямом падении лучей и при косвенном соответственно.

Данная функция реализована в шейдере as_Metal, где параметр r называется Face Reflectance, а g – Edge Reflectance. Оба принимают значения в диапазоне от 0 до 1. Также у шейдера важно отметить параметр roughness, который определяет шероховатость поверхности. При низком значении, например, 0.001 получается идеальное отражение, т.е. зеркальная поверхность.

4. Шейдер asStandardSurface

В данном шейдере поддерживается большое количество разных параметров:

1) Diffuse Parameters – диффузные свойства поверхности реализуются с помощью модели Орена-Наяра. [14] Такая же модель используется в Cycles в шейдере Diffuse BSDF;
2) Subsurface Parameters – свойства подповерхностного рассеивания реализуются с помощью BSSRDF (двунаправленная функция распределения отражательной способности поверхности рассеяния) [15];
3) Specular Parameters – регулирование отражательных свойств поверхности [3];
4) Refraction Parameters – параметры, контролирующие преломленные лучи [3];
5) Incandescence Parameters – параметры, позволяющие сделать материал источником света;
6) Transparency Parameters – параметр альфа канала, отвечает за видимость объекта;
7) Bump Parameters – параметр тонкой настройки микронормалей поверхности. Позволяет подключить карту нормалей.

Здесь приведены некоторые основные параметры, которые использовались в исследовании. Все далее описываемые материалы имеют похожую структуру, поэтому в них будут описаны только существенные различия и список принимаемых параметров. Все шейдеры appleseed реализованы на языке OSL и доступны по ссылке [13].

В рендере **Cycles**:

1. Diffuse BSDF

Данная модель [16] использует диффузную модель отражения Ламберта (англ. Lambertian) и Орен-Наяра (англ. Oren-Nayar) [14]:

$$L_r(\theta_r, \theta_i, \phi_r, \phi_i, \sigma) = \frac{c}{\pi}(A + B \max[0, \cos(\phi_r + \phi_i)] \sin\alpha \tan\beta)$$

где $A = 1.0 - 0.5\frac{\sigma^2}{\sigma^2+0.33}$;

$B = 0.45\frac{\sigma^2}{\sigma^2+0.09}$;

θ_r, ϕ_r – углы сферической системы координат вектора, направленного в сторону камеры;

θ_i, ϕ_i – углы сферической системы координат вектора, направленного в сторону источника света;

$\alpha = \max[\theta_r, \theta_i]$;

$\beta = \min[\theta_r, \theta_i]$;

c – альбедо, σ – коэффициент шероховатости.

При $\sigma = 0$, уравнение сводится к модели Ламберта: $L_r = \frac{c}{\pi}$

В Cycles реализован шейдер Diffuse BSDF с параметрами:

1) Color – вектор RGB
2) Roughness – коэффициент шероховатости σ

Также в качестве коэффициента Френеля здесь используется приближение Шлика [19]: $F(\theta) = F(0) + (1 - F(0))(1 - \cos\theta)^5$, где θ – угол падающей волны относительно нормали, $F(0)$ – отражение при нормальном падении.

2. Emission
   Шейдер используется для добавления освещения. Материал становится источником освещения [17].
   В качестве параметров принимает:

   1) Color – цвет источника;
   2) Strength – сила источника света в Ваттах.

3. Glossy BSDF
   Данный шейдер реализует отражающую свет поверхность, используя модель функции BRDF Кука-Торренса [18], [21].

   Нужную функцию распределения нормалей D можно выбрать из:

   1) Sharp – идеальное отражение, шероховатость никак не учитывается;
   2) GGX;
   3) Beckmann;
   4) Ashikhmin-Shirley.

В исследовании использовались модели GGX и Beckmann, которые были описаны ранее. Подробное описание всех моделей функции распределения можно найти в [20].

4. Glass BSDF

   Данная модель оптимизирована под стеклянную поверхность [22]. В ней используется BRDF Кука-Торренса [17] и есть возможность, так же как в предыдущем шейдере, выбрать используемое распределение для микрограней поверхности, чтобы регулировать степень шероховатости стекла:

   1) Sharp;
   2) GGX;
   3) Beckmann.

   Также через параметры задаются:

   1) Color – цвет стекла;
   2) Roughness – шероховатость, задается с помощью D;
   3) IOR (Index of refraction) – показатель преломления.

5. Principled BSDF

   Аналог asDisneyMaterial [33]. В данном шейдере соблюдаются те же принципы и присутствуют те же параметры, что и в оригинальной статье [11], [23].

6. Refraction BSDF

   Данный шейдер заставляет материал пропускать преломленные лучи [24].

   В нем так же можно выбрать модель распределения:

   1) Sharp;
   2) Beckman;
   3) GGX.

7. Transparent BSDF

   Альфа-канал материала [25]. Заставляет поверхность «растворяться», пропуская преломленные лучи, никак не влияя на их траекторию.

В рендере **LuxCore**:

1. Glass Material [26]

   Стеклянная поверхность. Параметры:

   1) IOR (Index of refraction) – показатель преломления;
   2) Reflection Color – цвет отраженных лучей;

3) Transmission Color – цвет преломленных лучей.

2. Matte Material
   Матовая поверхность, равномерно рассеивающая свет [27].
   Параметры:
   1) Diffuse Color – цвет поверхности;
   2) Sigma – шероховатость.

3. Matte Translucent Material
   Матовая поверхность, учитывающая подповерхностное рассеивание[28].
   Параметры:
   1) Reflection Color – цвет поверхности;
   2) Transmission Color – равномерный цвет преломленного света, проходящего через материал;
   3) Sigma – шероховатость.

4. Mirror Material
   Идеальная зеркально отражающая поверхность [29].
   Параметры:
   - Reflection Color – цвет зеркала. Позволяет изменять цвет отраженных лучей, направленных в камеру.

5. Disney Material
   Аналог шейдера asDisneyMaterial [33] и Principled BSDF [23]. [11]

6. Glossy Material
   Глянцевый материал [30]. Параметры:
   - Diffuse Color – диффузная компонента;
   - Specular Color and IOR – зеркальная компонента;
   - Roughness – шероховатость;
   - Absorption Color / Depth – цвет и глубина поглощения преломленных лучей;
   - Multibounce – мягкое рассеивание света на поверхности, имитирующее ткань или кожу.
7. Glossy Translucent Material
   Материал, смешивающий в себе диффузную, зеркальную компоненты и подповерхностное рассеивание [31].
   Параметры такие же, как и у предыдущего за добавлением:
   - Transmission Color – равномерный цвет преломленного света, проходящего через материал.

8. Metal Material

   Общая металлическая поверхность [32]. Пара метры:

   - Color/Fresnel Texture – цвет металла;
   - Roughness – шероховатость.

Цифровое изображение представляется в виде набора пикселей, выровненных по прямоугольной сетке. Для того чтобы вычислить дискретные значения пикселей в цифровом изображении, необходимо взять за образец исходную непрерывно определяемую функцию изображения. В PBR единственный способ получить информацию о функции изображения – это взять ее образец с помощью трассировки лучей. Изображение может быть создано только путем выборки значений функции изображения, которой является уравнение рендеринга, в конкретных позициях пикселей. Другими словами, это уравнение определяет цвет пикселя на изображении. Чем больше выборок для разных пикселей, тем лучше качество. Для решения этого уравнения существует множество подходов, однако наибольшую популярность получили методы Монте-Карло для численного решения интегралов. Рассмотрим алгоритмы, основанные на этом методе, реализованные в рендерах и используемые в исследовании.

Глава 1.3 Рендеры и алгоритмы трассировки

Алгоритм Path Tracing

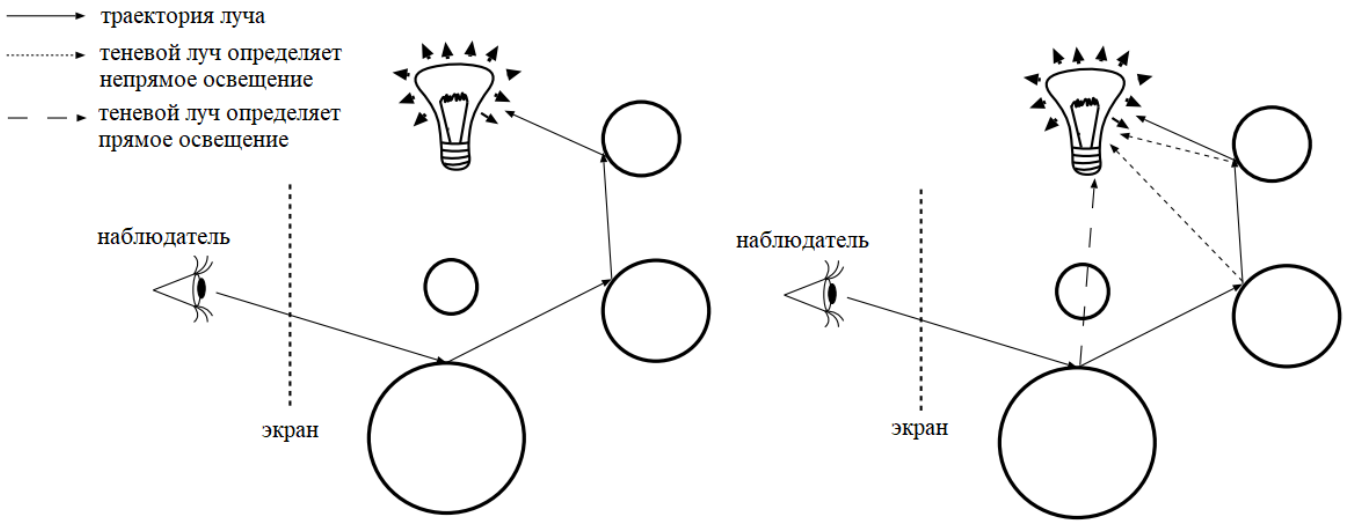


Рис. 6 Схема алгоритма трассировки пути без(слева) и с(справа) вычислением глобального освещения. [34]

Алгоритм трассировки пути был предложен Джеймсом Т. Каджия в той же работе, в которой было описано уравнение рендеринга. Он заключается в симуляции пути светового луча для каждого пикселя в изображении: когда луч пересекается с поверхностью, в псевдослучайном направлении выпускается отраженный луч и в обратном направлении теневой луч. Трассировка пути причисляется к Gathering type алгоритмам, которые начинают из положения камеры и «собирают» излучение посещенных точек.

1) Из точки на экране выпускается луч и далее ищется пересечение его с каким-либо объектом.
2) Если пересечений нет, то возвращаем свет окружения, другими словами, фон. Если пересечение произошло, то определяем альбедо объекта с как сумму его собственного излучения + прямое освещение источник света.
3) Функция `BRDFSampling` находит новое направление `newray`, используя различные оптимизационные методы, описанные далее. В частности, метод русской рулетки, позволяющий пропускать рекурсивный запуск `Trace(newray)`;
4) Рекурсивный вызов строит новый луч и возвращает энергетическую яркость объекта, с которым пересекся отраженный луч. Функция $w()$ возвращает число, которое определяет степень влияния этой энерг. яркости на первоначальный объект. Таким образом в данной строке происходит учет (сбор) влияния отражений луча от других объектов.

Псевдокод алгоритма в приложении А1

Алгоритм Stochastic Progressive Photon Mapping

SPPM (Stochastic Progressive Photon Mapping) – это модификация PPM (Progressive Photon Mapping), который, в свою очередь является модификацией Photon mapping. Рассмотрим по порядку каждый из них:

Метод фотонных карт (photon mapping) [5], [34] – двухэтапный алгоритм.

На первом этапе из источника света выпускаются фотоны – набор данных вида (p_i, ω_i, β_i), где p_i – вершины – точки на поверхности сцены, от которых он отражался;

ω_i – вектор направления из источника освещения в точку p_i;

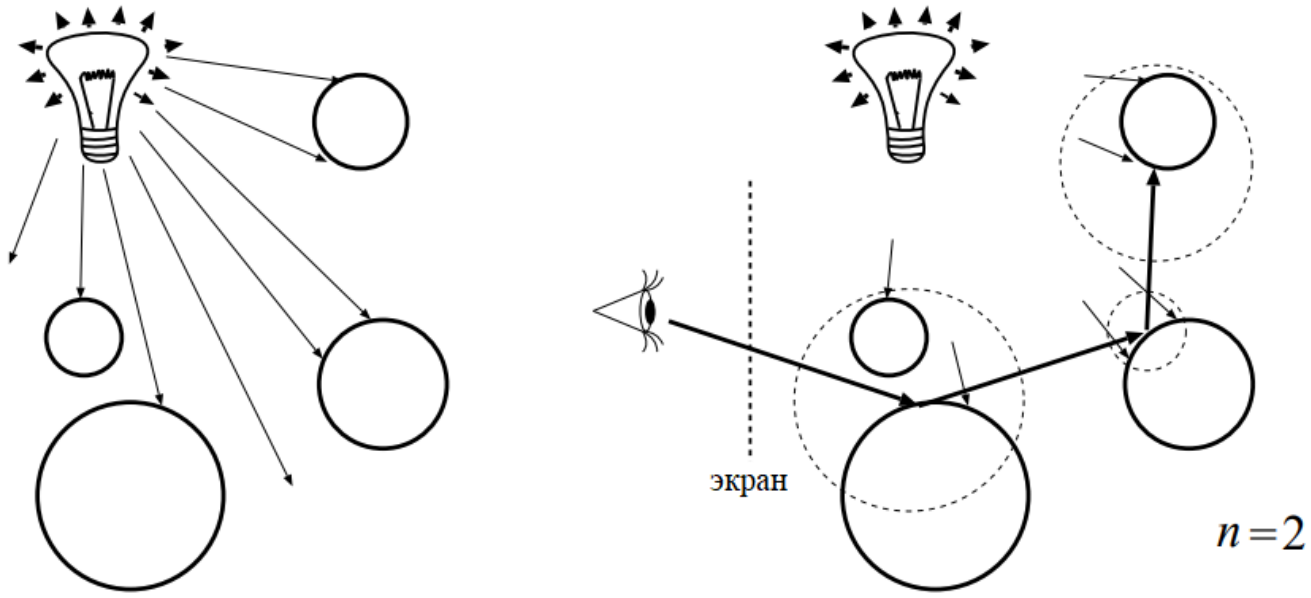
β_i – вес пропускной способности – характеристика, которая показывает сколько энергии остается у фотона после поглощения некоторого количества энергии в результате столкновения с поверхностью. Зависит от BSDF поверхности, энергетической яркости L_e и функции геометрии G.

Каждый такой набор сохраняется в кэше, и вся их совокупность называется фотонной картой.

С этим этапом связана проблема ограниченности памяти кэша: когда большую точность визуализации невозможно достичь, так как требуется больше фотонов, чем доступно памяти под их хранение.

Далее следует промежуточный оптимизационный этап построения kd-дерева, которое понадобится в дальнейшем для нахождения k ближайших «освещенных» точек.

На втором этапе лучи выпускаются из камеры в обратном направлении по путям лучей, которые достигли камеры на первом этапе. В каждой вершине p_i в окрестности некоторого радиуса, который задается как параметр, ищутся соседние фотоны для оценки непрямого освещения.



Первый этап. Создание фотонной карты

Второй этап. Сбор информации в некотором радиусе

Рис. 10 Схема этапов метода фотонных карт. [34]

Прогрессивный метод фотонных карт (Progressive Photon Mapping) [5] – реструктуризация метода фотонных карт. Теперь на первом этапе испускаются лучи из каждого пикселя камеры. И для тех лучей, что достигают диффузную поверхность, либо напрямую, либо отражаясь или преломляясь при столкновении отражающей или прозрачной средой, в кэше сохраняется соответствующим набором информации о них: геометрическая информация о поверхности, диффузный цвет, учет влияния на эти данные другой поверхности (в случае прохождения через неё) и т.д. Этот массив называется видимыми точками (англ. visible points).

Далее, на втором этапе, фотоны испускаются из источника освещения и каждый из них при столкновении с поверхностью вносит свой вклад в энергетическую яркость ближайших видимых точек. Причем прямое освещение можно рассчитать обычными алгоритмами, например, трассировкой пути. Для преломленных лучей перед этим необходимо проследить путь до следующей вершины p_i.

В данном способе также есть проблема памяти, которая связана с ограниченностью хранилища видимых точек. Поэтому для изображений с высоким разрешением или для тех, что требуют большого количества проходов, например, при создании эффекта размытости (англ. blur) или при большой глубине резкости, память все еще может быть ограничением.

Стохастический прогрессивный метод фотонных карт (Stochastic Progressive Photon Mapping) – модификация PPM – не имеет никаких ограничений по памяти. Алгоритм итеративно повторяет этапы PPM. На первом этапе, так же, как и в PPM, лучи испускаются из камеры, но по более упрощенной модели, экономящей память (например, испускается только один луч для каждого пикселя). Далее повторение второго этапа PPM. После этого видимые точки сбрасываются и создается новый массив с другим набором данных, процесс повторяется.

Также постепенно с увеличением кол-ва проходов и, тем самым, увеличением кол-ва фотонов, содержащих информацию о точке поверхности, уменьшается окрестность поиска фотонов. Уменьшая радиус, мы гарантируем, что фотоны, которые будут использоваться, будут ближе к точке и, таким образом, будут способствовать более точной оценке распределения падающего излучения.

Псевдокод алгоритма в приложении А2

Алгоритм Bidirectional path-tracing (BDPT)

BDPT сначала последовательно строит путь начиная с точки p_0 на камере. Следующая вершина, p_1, находится путем вычисления первого пересечения вдоль луча камеры. Другая вершина находится путем семплирования BSDF в точке p_1 и

трассировки луча, чтобы найти точку p_2, и так далее. Результирующий путь из t вершин: p_0, ... p_{t-1}.

Аналогично, начиная с точки на источнике света q_0 (используя соответствующие характеристики поверхности (BSDF) в каждой вершине), создается световой подпуть из s вершин: q_0,... q_{s-1}.

Полный путь есть соединение вершин подпутей:

$$P = q_0, \dots, q_{x-1}, p_{y-1}, \dots, p_0$$

где x ≤ s и y ≤ t. [5]

Можно выделить три способа улучшения алгоритма:

1) Подпути могут быть использованы повторно. Если задан путь P, то соединяя всевозможными способами различные комбинации префиксов $q_i, 1 < i < s$ и $p_j, 1 < j < t$, можно построить множество уникальных путей длины $2 < l < s + t$.

2) Нельзя пытаться соединять пути, использующие только одну вершину, относящуюся к одному из подпутей. Предпочтительнее построить такие пути с применением оптимизированных функций дискретизации, предоставляемых камерой и источниками света; для источников света, например, таковыми являются техники прямого освещения.

3) Можно оценивать различные стратегии для генерации путей определенной длины, потому что очевидно, что некоторые стратегии, которые хороши для создания одних видов траекторий, могут быть плохи для других.

Множественная выборка по важности (англ. Multiple importance sampling) может быть применена для оценивания стратегий и использования их там, где они лучше всего подходят. Это применение MIS имеет решающее значение для эффективности BDPT.

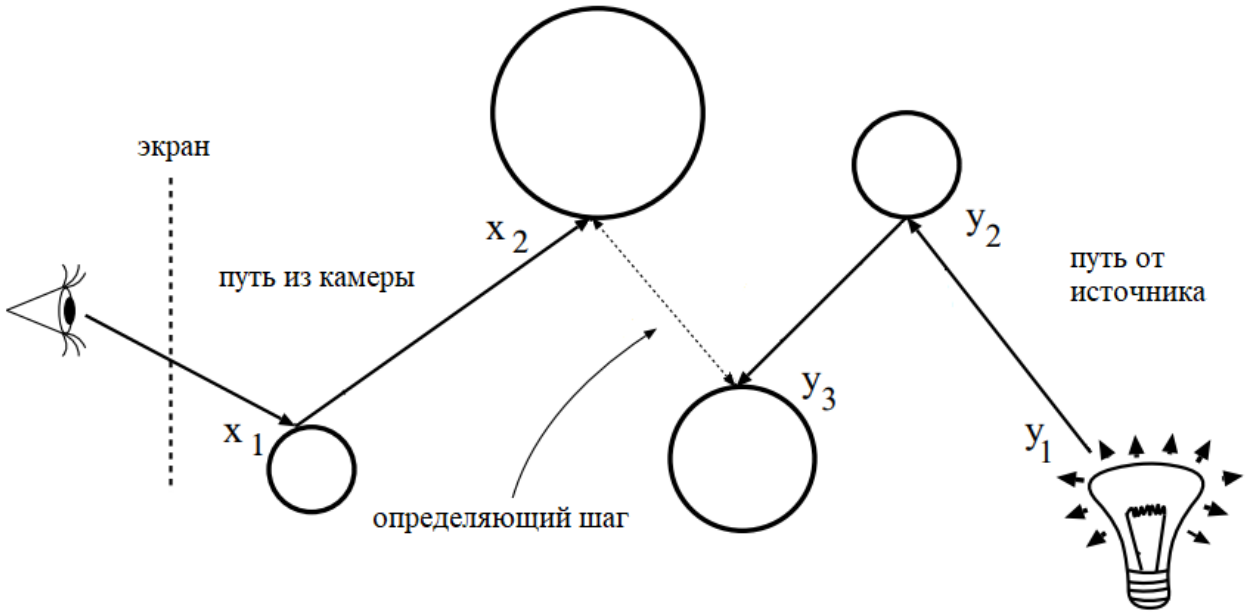


Рис. 7 Схема алгоритма двунаправленной трассировки пути с единичным детерминированным шагом. [34]

Оптимизационные методы

Для каждого алгоритма существуют способы ускорить его работу. Общая идея состоит в том, чтобы выбирать те лучи, которые вносят наибольший вклад в изображение. Рассмотрим некоторые основные способы, которые упоминались выше.

Metropolis Sampling

Это техника получения набора выборок для известной неотрицательной функции $f(x)|\ f: \mathbb{R}^n \rightarrow \mathbb{R}$. После определения первого случайного значения $X_0 \in \mathbb{R}^n$, X_i выбирается с помощью случайной мутации x', для которой определены условия. Если они выполняются, то $X_i = x'$, иначе $X_i = X_{i-1}$. Она была описана в статье [39].

Выборка по важности (англ. Importance sampling)

Отражение лучей в полностью случайном направлении может привести к большим и нецелесообразным затратам, поэтому имеет смысл выбирать направление в зависимости от возвращаемого значения функции BRDF. Если оно близко к нулю, то скорее всего такой луч не привнесет в сцену заметных изменений. И наоборот, в направления, которые приводят к большим значениям BRDF, следует выпускать больше лучей.

Приближение Монте-Карло (Monte-Carlo estimator) – аппроксимация уравнения рендеринга:

$$F_N = \frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{p(X_i)}$$

где $X_i \in [0,1]$ – случ. величины, распределенные в соответствии с плотностью $p(x)$;

$p(x)$ – функция плотности вероятности (англ. Probability density function),

Причем должно выполнятся $\forall x$: $|f(x)| > 0 \Rightarrow p(x) \neq 0$;

N кол-во случайных величин X_i ;
$f(x)$ – подынтегральное выражение. [5]
Выборка по важности – метод уменьшения дисперсии, которая позволяет оценке сходиться быстрее путем контролирования PDF, а значит и выбираемых X_i.
Чем больше $p(x)$ похожа на $f(x)$, тем точнее оценка.

Множественная выборка по важности (англ. Multiple importance sampling)

Применяется для уменьшения дисперсии интегралов вида $\int f(x)g(x)dx$.

Целью является получение хотя бы одной функции плотности вероятности, которая будет похожа на форму подынтегральной функции. [5]
Пусть есть PDF p_f и p_g функций $f(x)$ и $g(x)$ соответственно.
Тогда новое приближение Монте-Карло имеет вид:

$$\frac{1}{n_f}\sum_{i=1}^{n_f}\frac{f(X_i)g(X_i)\omega_f(X_i)}{p_f(X_i)}+\frac{1}{n_g}\sum_{j=1}^{n_g}\frac{f(Y_j)g(Y_j)\omega_g(Y_j)}{p_g(Y_j)}$$

где n_f – кол-во случайных величин, распределенных в соответствии с плотностью p_f;

n_g – кол-во случайных величин, распределенных в соответствии с плотностью p_g;

ω_f, ω_g – весовые функции, которые должны подбираться так, чтобы мат. ожидание данной оценки равнялось подынтегральной функции $f(x)g(x)$, а также учитывать всевозможные способы выбора сл. вл. X_i и Y_j.

Примером такой функции может быть так называемая балансировочная эвристика (англ. the balance heuristic) [35] :

$$\omega_k(x)=\frac{n_k p_k(x)}{\sum_i n_i p_i(x)}$$

Русская рулетка и расщепление (англ. Russian roulette and Splitting)

Это два связанных метода, призванных улучшить сходимость метода Монте-Карло за счет повышения вероятности того, что каждая выборка внесет существенный вклад в результат. [5]

Русская рулетка позволяет пропускать вычисления, которые заведомо не принесут или привнесут, но очень мало результата. Например, если в уравнении рендеринга для некоторого множества направлений ω_i значение функции $f_s(p, \omega_o, \omega_i)$ близко или равно 0, то, очевидно, отслеживание этого луча не имеет смысла и его необходимо пропустить. Также русская рулетка позволяет не учитывать значения, которые падают при очень маленьком угле относительно горизонта ($|\cos\theta| = 0 + \varepsilon$).

Расщепление – метод, который распределяет количество пущенных лучей в зависимости от вклада, который эти лучи могут привнести. Например, если для приближения интеграла взято 100 случайных величин, то необходимо проследить траекторию преломленных и отраженных в направлении камеры лучей. Используя

обычный метод, алгоритм просчитывал бы 200 лучей в целом. Но 100 отраженных лучей – это избыток.

С помощью расщепления приближение для такого случая можно заменить на следующее:

$$F_{NM} = \frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}\frac{L(x_i, y_i,\ \omega_{i,j})}{p(x_i, y_i)p(\omega_{i,j})}$$

где $\omega_{i,j}$ – направление на источник света
(x_i, y_i) – координаты пикселя на изображении.

В исследуемых рендерах реализованы следующие алгоритмы:

1. **appleseed**
   Алгоритм Path Tracing и Stochastic Progressive Photon Mapping
   Для просчета прямого освещения можно выбрать способ просчета:
   - с помощью Path Tracing
   - с помощью SPPM
   - без просчета

   В алгоритме трассировки пути есть возможность выбрать способ итерирования:

   - По умолчанию – количество итераций задается пользователем.
   - Адаптивный семплинг – количество итераций задается пользователем. Также задается минимальный уровень шума, при котором определяется количество дополнительных семплов для определенного пикселя, минимальное и максимальное кол-во которых также задается пользователем.

   В исследовании использовалась версия 2.2
2. **Cycles**
   Алгоритм Path Tracing
   А также Branched path tracing, для которого можно указать точное количество отражений лучей от определенного типа поверхности.
   В исследовании использовалась версия Blender 2.82

3. **LuxCore**
   Алгоритм Path Tracing и Bidirectional path-tracing
   Также в этом рендере есть технология PhotonGI cache [36], которая включает в себя 3 функции:
   - Direct light cache – кеш прямого освещения;
   - Indirect light cache – кеш непрямого освещения;

- Caustic light cache – кеш каустик.

Данная технология основана на методе фотонных карт [34]. В комбинации с Path tracing данный алгоритм хорошо подходит для просчета сложного освещения.

В исследовании использовалась версия 2.4

Глава 1.4 Способ сравнения качества изображений

Основная метрика для сравнения изображений – PSNR (peak signal-to-noise ratio) – пиковое отношение сигнала к шуму. [37] PSNR определяется через среднеквадратическую ошибку (англ. mean squared error):

$$MSE = \frac{1}{m \cdot n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} |I(i,j) - Ihat(i,j)|^2$$

где I и $Ihat$ изображения размера $m \times n$, последнее из которых считается зашумленным приближением другого.

Для цветных изображений с тремя цветовыми каналами (RGB) MSE – это сумма всех квадратов разностей значений, разделенная на размер изображения и на 3. Кроме того, для цветных изображений изображение преобразуется в другой цветовой формат – YCbCr, и PSNR сообщается по каждому каналу.

$$PSNR = 10\ log_{10}\left(\frac{MAX_I^2}{MSE}\right)$$

где MAX_I – максимальное значение яркости пикселя изображения.

В данном исследовании используется функция модуля `skimage.metrics` языка Python: `peak_signal_noise_ratio(image_true, image_test, data_range=None)`[38]

Для конвертации изображения и одновременно нормирования яркости используется код [приложение А3] из нескольких функций модуля `cv2.`

Для исследования распараллеливания в первых двух экспериментах алгоритм поочерёдно запускался на 6 и 12 потоках.

Эксперименты

Все эксперименты проводились на CPU AMD Ryzen 5 3600 3.60 ГГц.
Операционная система Windows 10.

Эксперимент 1. Стеклянный стакан

Разрешение изображения: 1920x1080. [приложение В1]

1. В appleseed используется алгоритм SPPM
   Для обсчета прямого освещения используется SPPM
   Кол-во испускаемых лучей из источника: 10M;
   Радиус окрестности: 0.1;
   Начальное кол-во отскоков возможных отражений до ограничения семплирования с помощью русской рулетки: 6
   Кол-во итераций (семплов): 25, 52, 156, 370.
2. Cycles
   Используется алгоритм Path Tracing.
   Максимальное кол-во отражений/преломлений лучей (за 1 итерацию) испускаемых от камеры от:
   - Диффузной поверхности – 4;
   - Глянцевой поверхности – 50;
   - Пропускающей поверхности – 40;

   Кол-во итераций (семплов): 17, 32, 89, 195.
3. В LuxCore
   Используется алгоритм Path Tracing + Caustic light cache;
   Кол-во испускаемых лучей из источника: 10M;
   Радиус окрестности: 0.01;
   Максимальное кол-во отражений фотона, выпущенного из источника света: 16;
   Максимальное кол-во отражений лучей, испускаемых от камеры от:
   - Диффузной поверхности – 16;
   - Глянцевой – 36;
   - Зеркальной – 38;

Кол-во итераций (семплов): 148, 292, 854, 1898.

Эксперимент 2. Стеклянные модели бутылок и зеркальная поверхность.

Разрешение изображения: 2000x1700 px. [приложение В2]

1. appleseed
   Максимальное кол-во отражений/преломлений испускаемых лучей:
   Диффузной поверхности – 8;
   Глянцевой поверхности – 8;
   Пропускающей поверхности – 8;
   Кол-во итераций (семплов): 7, 14, 27, 54.
2. Cycles

Максимальное кол-во отражений/преломлений испускаемых лучей:
Диффузной поверхности – 16;
Глянцевой поверхности – 6;
Пропускающей поверхности – 8;
Кол-во итераций (семплов): 36, 72, 139, 280.

3. LuxCore

Максимальное кол-во отражений/преломлений испускаемых лучей:
Диффузной поверхности – 4;
Глянцевой(отражающей) поверхности – 12;
Пропускающей поверхности – 12;
Кол-во итераций на (семплов): 88, 173, 335, 676.

Эксперимент 3. Полупрозрачная штора.

Разрешение изображения: 1120x1540 px. [приложение В3]

1. appleseed

Используется алгоритм трассировки пути с двумя проходами для оценки прямого освещения. Просчет каустиков отключен.
Максимальное кол-во отражений/преломлений испускаемых лучей – 10;
Диффузной поверхности – 10;
Глянцевой поверхности – 8;
Пропускающей поверхности – 8;
Кол-во шагов адаптивного семплера: 5, 10, 20, 40.
Параметры адаптивного семплера заданы по умолчанию.

2. Cycles

Используется алгоритм трассировки пути
Максимальное кол-во отражений/преломлений испускаемых лучей – 16;
Диффузной поверхности – 14;
Глянцевой поверхности – 4;
Пропускающей поверхности – 16;
Прозрачной поверхности – 16.
Кол-во итераций на (семплов): 1333, 2624, 5172, 10486.

3. LuxCore

Для данного эксперимента было получено два набора экспериментов алгоритмами Path tracing и Bidirectional Path tracing

В алгоритме Path tracing:
Максимальное кол-во отражений/преломлений испускаемых лучей – 16;
Диффузной поверхности – 16;
Глянцевой поверхности – 4;
Пропускающей поверхности – 6;
Кол-во итераций на (семплов): 1058, 2084, 4106, 8326.

В алгоритме Bidirectional Path tracing:
Максимальное кол-во отражений луча:

- Выпущенного из камеры – 12,
- Выпущенного из источника света – 16.

Также используется оптимизационный алгоритм семплирования Metropolis.

- Процент мутации – 40%;
- Максимальное кол-во мутаций одного луча – 512.

Кол-во итераций (семплов): 401, 789, 1556, 3145.

Эксперимент 4. Модель световода.

Разрешение изображения: 1600x900 px. [приложение В4, В5]

1. В appleseed используется алгоритм SPPM

Для обсчета прямого освещения используется SPPM.

Кол-во испускаемых лучей из источника: 1M;

Радиус окрестности: 0.1;

Начальное кол-во отскоков возможных отражений до ограничения семплирования с помощью русской рулетки: 12

Кол-во итераций (семплов): 177, 354, 707, 1413.

2. Cycles

Используется алгоритм Branched Path Tracing.

Максимальное кол-во отражений/преломлений лучей (за 1 итерацию) испускаемых от камеры от:

- Диффузной поверхности – 64;
- Глянцевой поверхности – 50;
- Пропускающей поверхности – 50;

Кол-во итераций (семплов): 40, 80, 160, 320.

3. LuxCore

Используется алгоритм Bidirectional Path-tracing.

Максимальное кол-во отражений луча:

- Выпущенного из камеры – 16,
- Выпущенного из источника света – 16.

Также используется оптимизационный алгоритм семплирования Metropolis.

- Процент мутации – 20%;
- Максимальное кол-во мутаций одного луча – 512.

Кол-во итераций (семплов): 599, 1198, 3058, 6114.

Эксперимент 5. Модель бассейна.

Разрешение изображения: 1700x1200 px. [приложение В6]

1. appleseed

Используется алгоритм Path Tracing с двумя проходами для оценки прямого освещения. Просчет каустиков включен.

Максимальное кол-во отражений/преломлений испускаемых лучей – 10;

- Диффузной поверхности – 3;

- Глянцевой поверхности – 10;
- Пропускающей поверхности – 8;

Кол-во итераций (семплов): 7, 17, 34, 65.

2. Cycles

Используется алгоритм Branched Path Tracing.

Максимальное кол-во отражений/преломлений лучей (за 1 итерацию) испускаемых от камеры от:

- Диффузной поверхности – 8;
- Глянцевой поверхности – 14;
- Пропускающей свет поверхности – 14;

Кол-во итераций (семплов): 24, 60, 120, 229.

3. LuxCore

Используется алгоритм Bidirectional Path-tracing.

Максимальное кол-во отражений луча:

- Выпущенного из камеры – 16,
- Выпущенного из источника света – 16.

Также используется оптимизационный алгоритм семплирования Metropolis.

- Процент мутации – 60%;
- Максимальное кол-во мутаций одного луча – 512.

Кол-во итераций (семплов): 100, 250, 500, 962.

3 Сравнение рендеров, выводы

Для определения качества распараллеливания проводятся испытания сначала на 6 потоках, потом на 12. Процентное соотношение вычислялось по формуле:

$$t_{\%} = \frac{time_{12}}{time_{6}} \cdot 100$$

Эксперимент 1

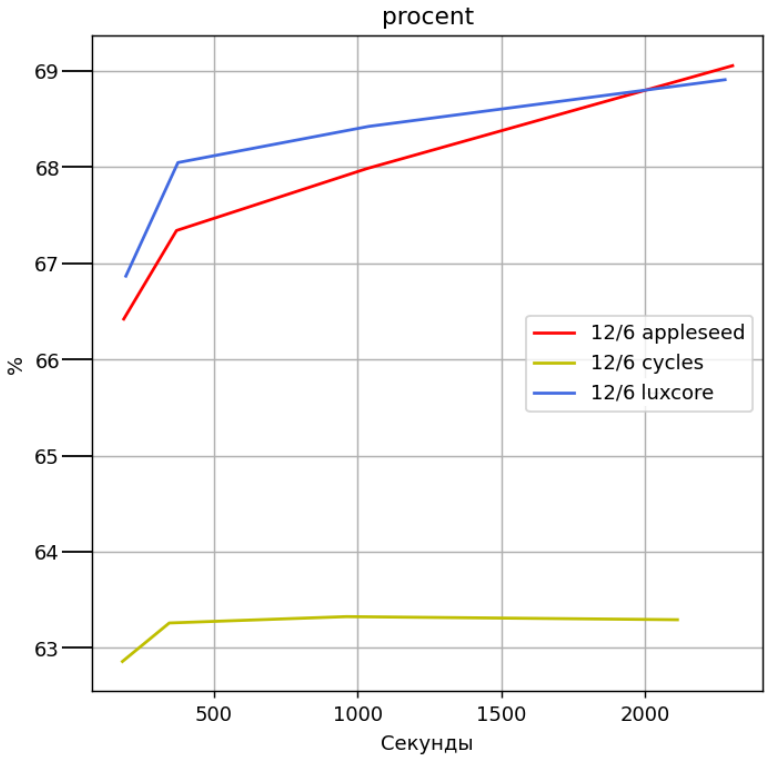


Рис. 1 Процентное соотношение скорости от количества потоков

Из данного на Рис. 1 графика видно, что рендеры appleseed и luxcore улучшают время примерно на 68% для алгоритмов sppm и path tracing + caustic light cache при увеличении кол-ва потоков в 2 раза. Рендер Cycles, который использует path tracing, улучшает скорость на ~63%.

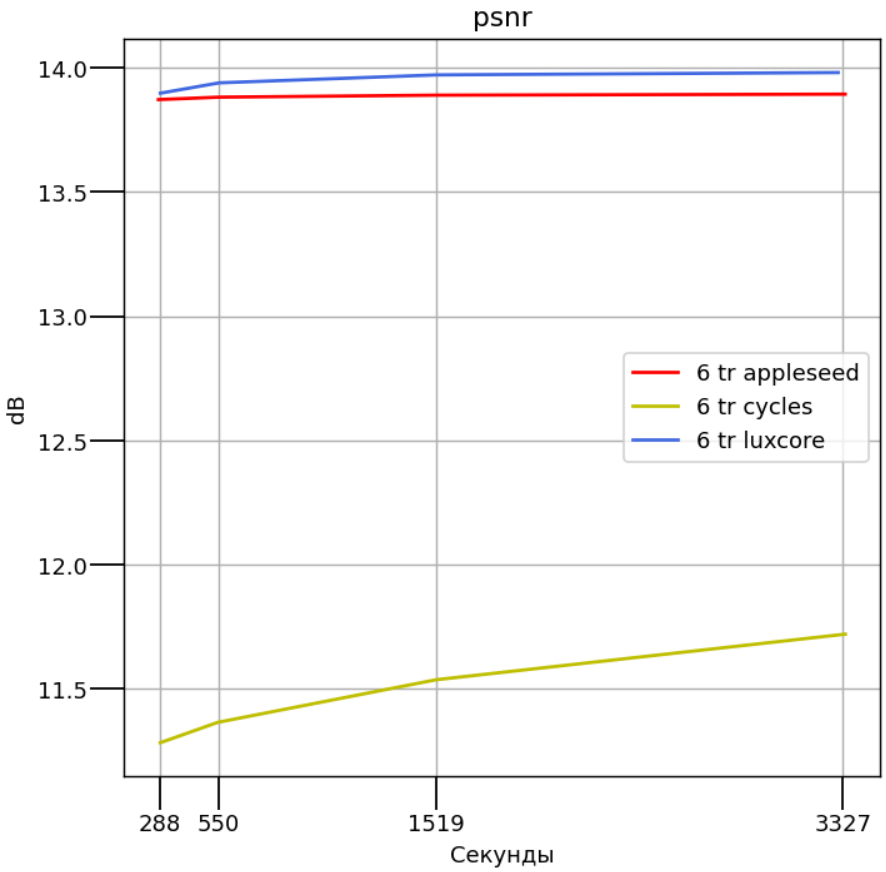


Рис.2 Зависимость качества от времени работы рендеров.
Время при 6 потоках

В данном видно, что luxcore и appleseed показывают примерно одинаковый результат результат по схожести с эталонным изображением. Cycles показывает много более низкий результат за то же время. Это объясняется тем, что алгоритм Path tracing плохо подходит для отрисовки непрямого освещения и каустиков. Также важно обратить внимание на сами графики, а именно на изменения psnr в зависимости от времени. Средняя скорость роста качества:

0. appleseed = 0.007789704187192233
1. cycles = 0.14529101277787385
2. luxcore = 0.03280095439115686

Можно заметить, что luxcore и appleseed имеют небольшое изменение psnr относительно времени. Это говорит о том, что данные рендеры достигли своего достигли пика качества, когда последующие итерации не вносят значительного результата в изменение изображения. Из этого можно сделать вывод, что они работают быстрее path tracing в cycles в данном эксперименте. Наиболее высокое качество было достигнуто рендером luxcore.

Эксперимент 2

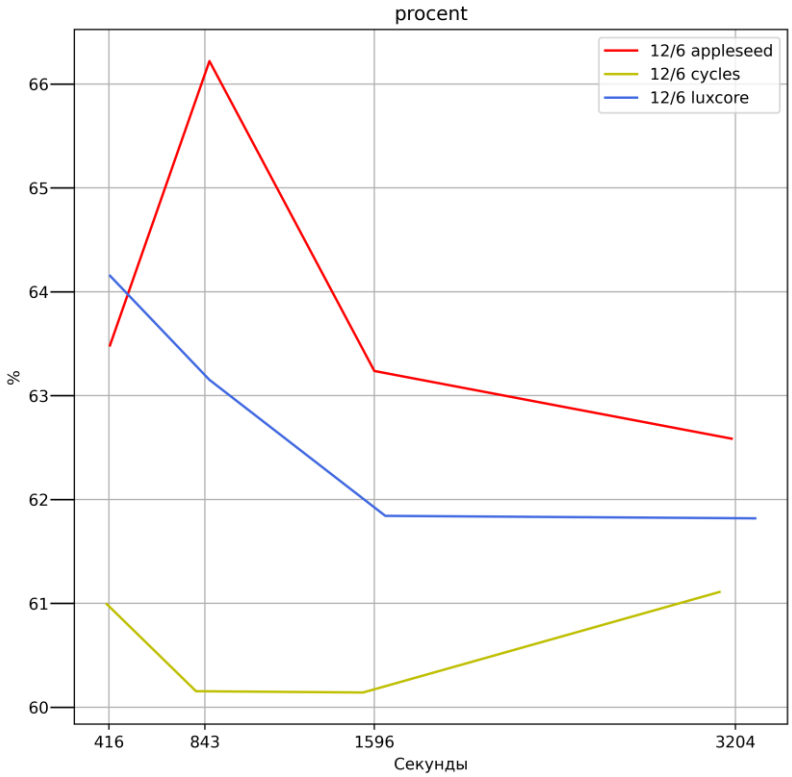


Рис. 3 Процентное соотношение скорости от количества потоков

Для отрисовки данной сцены во всех рендерах использовались алгоритмы трассировки пути. Из данного эксперимента можно сделать аналогичный с прошлым экспериментом вывод – среднее увеличение скорости этого алгоритма составляет 60 %.

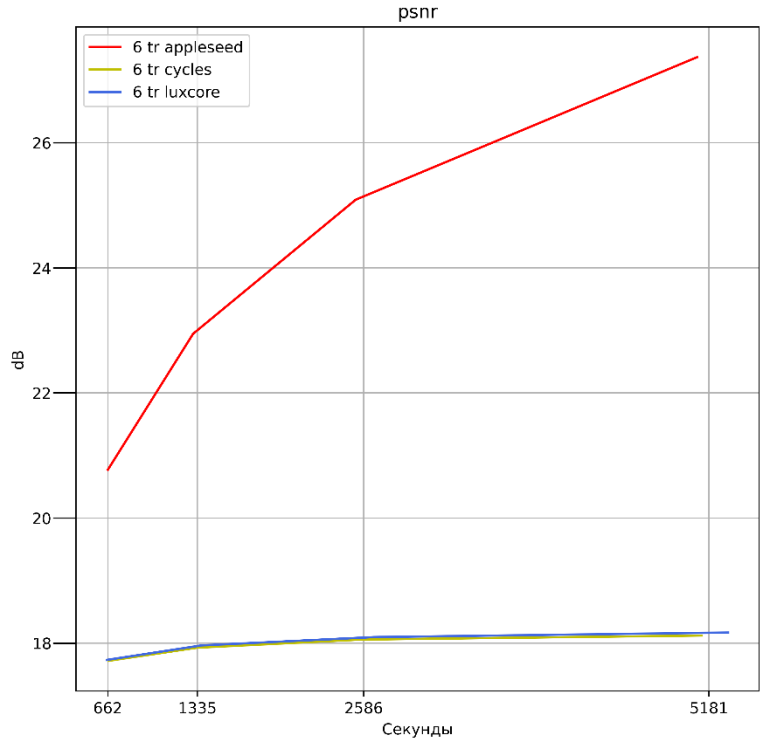


Рис. 4 Зависимость качества от времени работы рендеров. Время при 6 потоках

Из Рис. 4 видно, что appleseed показывает наибольшую схожесть с эталоном, но следует учесть, что на такой результат влияют:

1. первое, то что полностью идентичный результат в принципе невозможно получить разными рендерами.
2. в качестве эталонного изображения взят результат модели того же appleseed, но за неограниченное время.

Видно, что примерно одинаковый результат получают рендеры cycles и luxcore. Также важно обратить внимание на скорость роста графиков.

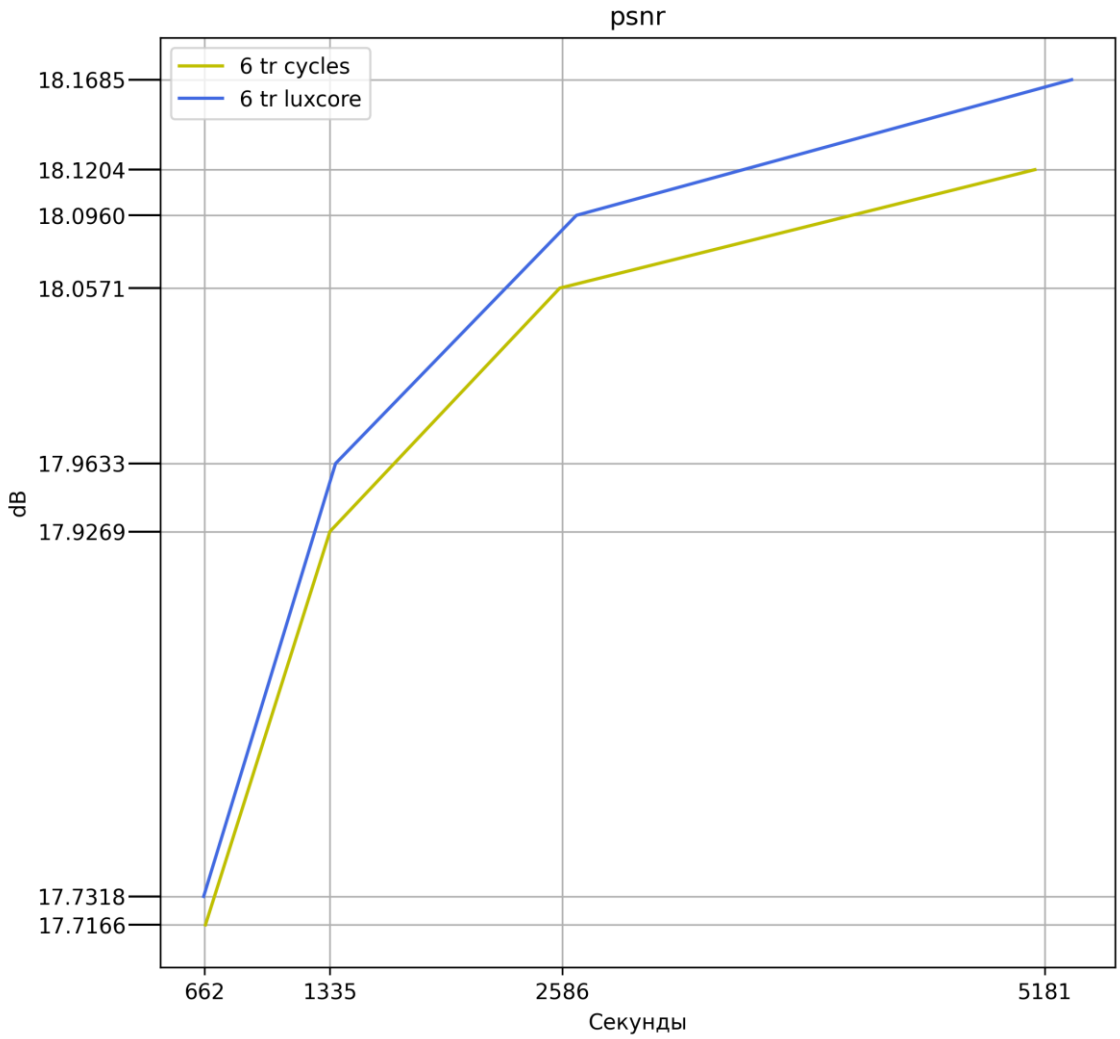


Рис.5 Зависимость качества от времени работы рендеров luxcore и cycles. Время при 6 потоках

Как можно видеть рис. 4, скорость роста у рендеров LuxCore и cycles много меньше, чем у appleseed. В данном случае это говорит о том, что рендеры сходятся к другому изображению, отличному от эталона. На рис. 5 видно, что рендеры cycles и LuxCore за одинаковый промежуток времени дают близкие по качеству друг к другу результаты. Отсюда можно сделать вывод, что для данного типа моделей для быстрого результата оптимально выбирать cycles или LuxCore, а для максимально реалистичного – appleseed.

Эксперимент 3

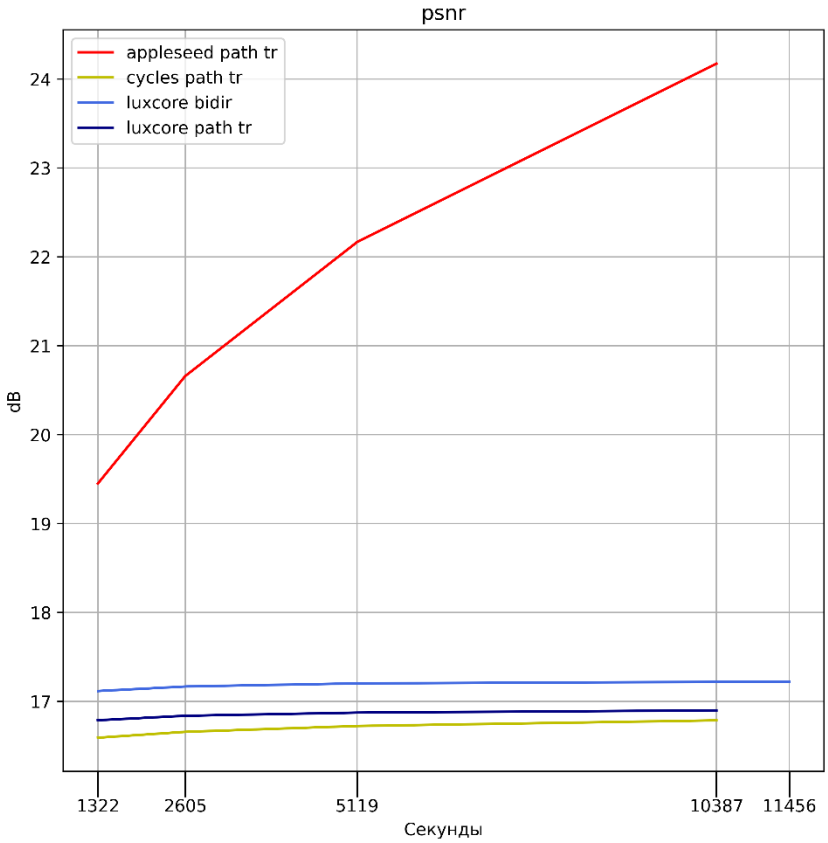


Рис.6 Зависимость качества от времени работы рендеров.

В данном эксперименте были протестированы алгоритмы трассировки пути каждого рендера, а также двунаправленная трассировка рендера luxcore. Из рис. 6 видно, что наибольшую реалистичность показал рендер appleseed. Это опять же связано с тем, что эталонное изображение было получено с помощью этого рендера. Остальные алгоритмы показали приблизительно схожие друг с другом результаты.

На рис. 7 видно, что среди них за то же время лучшее качество продемонстрировал алгоритм двунаправленной трассировки пути. Из этого можно сделать вывод, что для отрисовки данного типа сцен оптимально выбирать appleseed или luxcore.

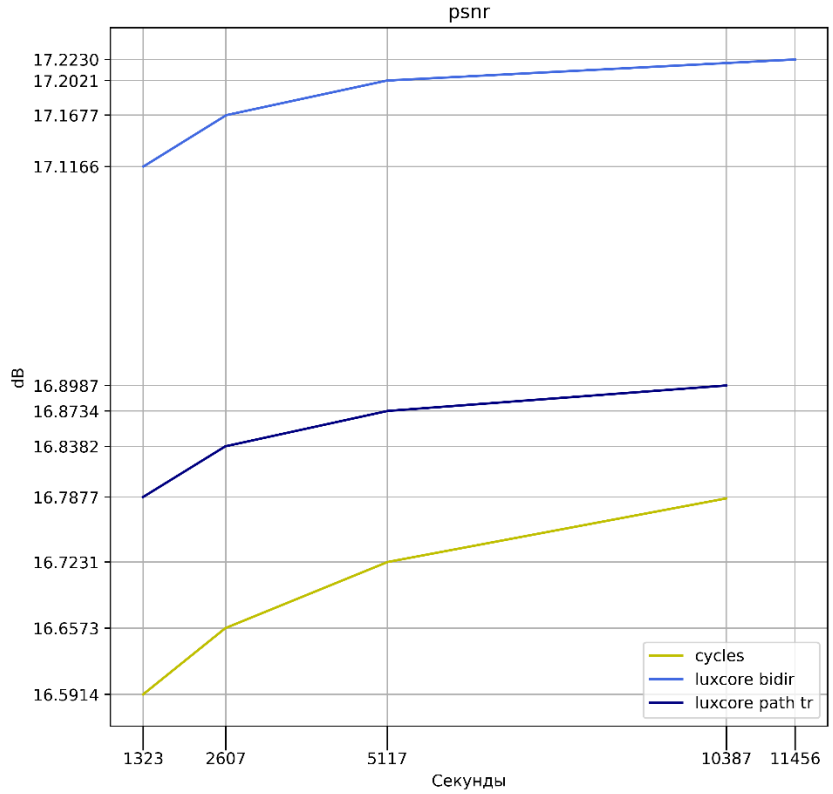


Рис.7 Зависимость качества от времени работы рендеров cycles и luxcore.

Эксперимент 4

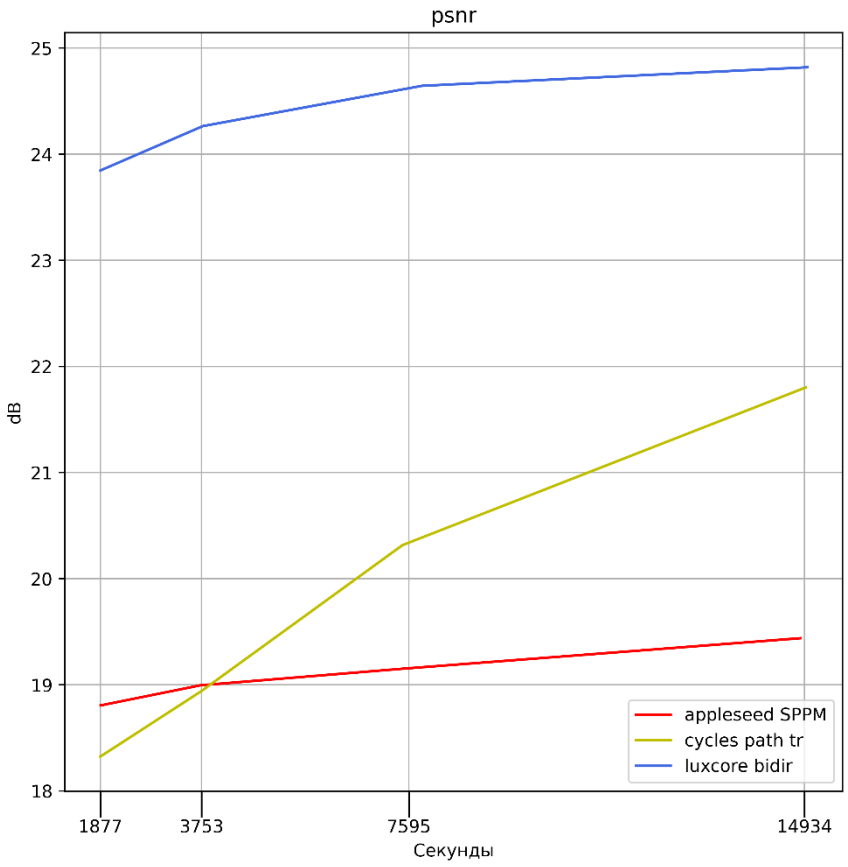


Рис.8 Зависимость качества от времени работы рендеров.

В данной сложной сцене источник освещения находился за стеной, что ограничивало его влияние на окружающее освещение. Световые лучи двигались по стеклянной трубке с показателем преломления IOR = 1.9. [Приложение B5] Наилучшее качество показал алгоритм двунаправленной трассировки пути рендера luxcore. Трассировка пути, реализованная в Cycles, как видно из рис. 8 постепенно приближается к эталону. Рендер appleseed использовал SPPM, который с самого начала выдал сильно зашумленный результат и с течением времени смог лишь незначительно его улучшить. Выводы из данного эксперимента очевидны.

Эксперимент 5

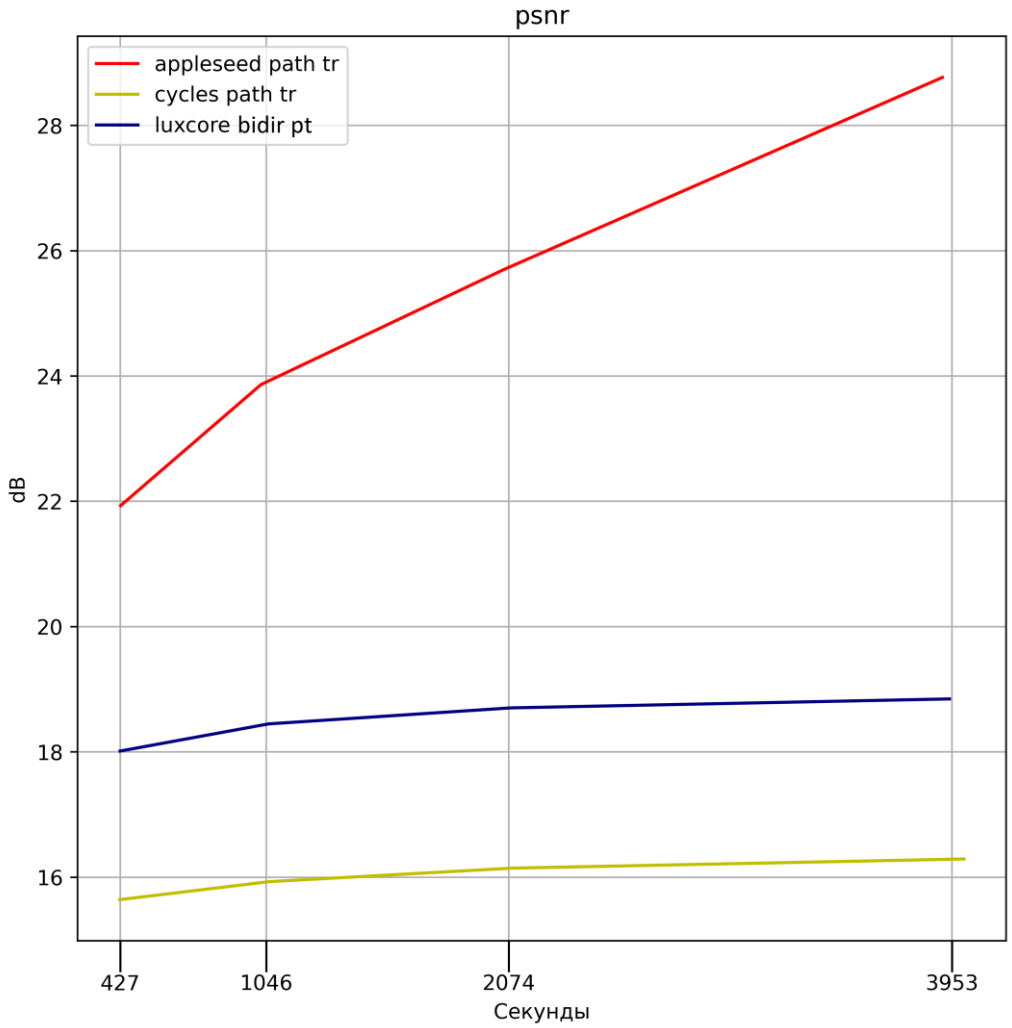


Рис. 9 Зависимость качества от времени работы рендеров.

В данном эксперименте можно сделать выводы, аналогичные выводам второго эксперимента. Из данного графика видно, что рендер appleseed, используя трассировку пути, получил наиболее реалистичный результат, схожий с эталоном.

Заключение

В представленном исследовании произведено сравнение рендеров appleseed, Cycles и Luxcore по качеству построенных изображений трёхмерных сцен. Результаты их работы сравнивались при ограниченном времени работы и была рассмотрена динамика улучшения качества изображения при увеличении времени работы.

В большинстве тестов наиболее фотореалистичным являлся рендер appleseed, который в условиях ограниченного времени давал наиболее фотореалистичное изображение. Рендеры Cycles и LuxCore демонстрировали схожие результаты: они давали изображения приблизительно равного качества в условиях ограниченного времени, а с течением времени качество изображения менялось мало.

Таким образом для моделей в которых преобладает непрямое освещение, лучше использовать двунаправленный алгоритм, в котором лучи трассируются как из источников освещения, так и из камеры – SPPM в appleseed или bidirectional path tracing в LuxCore. Для простых моделей с преобладанием прямого освещения для наиболее качественного результата следует использовать трассировку пути, реализованную в рендере appleseed, если же в приоритете является время рендеринга, то лучшим выбором будет рендер Cycles.

Список использованных источников

1. Сивухин Д. В. Общий курс физики. Учеб. пособие: Для вузов. В 5 т. Т. IV. Оптика. — 3-е изд., стереот. – М.: ФИЗМАТЛИТ. – 2005. – 792 с. – ISBN 5-9221-0228-1.
2. Электронный учебник по курсу "Основы оптики" URL: http://aco.ifmo.ru/el_books/basics_optics/ (дата обращения 10.05.2020)
3. Bruce Walter, Stephen R. Marschner, Hongsong Li, Kenneth E. Torrance Microfacet Models for Refraction through Rough Surfaces Proceedings of the 18th Eurographics Conference on Rendering Techniques. – 2007. – 195-206 P. doi:10.2312/EGWR/EGSR07/195-206.
4. Kajiya, James T. The rendering equation // Siggraph 1986. – 1986 – P. 143–150. – doi:10.1145/15922.15902
5. Matt Pharr, Wenzel Jakob, Greg Humphreys Physically Based Rendering: From Theory To Implementation. – 2018. – 1226 P.
6. Альбедо в реалистичной визуализации URL: https://render.ru/ru/m.trofimov/post/11166 (дата обращения 28.04.2020)
7. Marco Alamia Physically Based Rendering URL: http://www.codinglabs.net/article_physically_based_rendering.aspx (дата обращения 5.05.2020)
8. Physically Based Shading in Theory and Practice // SIGGRAPH 2013 Course URL: https://blog.selfshadow.com/publications/s2013-shading-course/ (дата обращения 5.05.2020)
9. Brian Karis Specular BRDF Reference URL: http://graphicrants.blogspot.com/2013/08/specular-brdf-reference.html (дата обращения 18.05.2020)
10. Bruce G. Smith Geometrical shadowing of a random rough surface // IEEE Transactions on Antennas and Propagation. – 1967. – 668-671 P. doi:10.1109/TAP.1967.1138991
11. Brent Burley Physically-Based Shading at Disney // SIGGRAPH 2012 Course: Practical Physically Based Shading in Film and Game Production. – 2012. doi:10.1145/2343483. – URL: https://blog.selfshadow.com/publications/s2012-shading-course/ (дата обращения 12.05.2020)

12. Ole Gulbrandsen Artist Friendly Metallic Fresnel // Journal of Computer Graphics Techniques (JCGT). – 2014. – Vol. 3, No. 4. – 64-72 P.
13. Appleseedhq, appleseed shaders URL: https://github.com/appleseedhq/appleseed/tree/master/src/appleseed.shaders/src/appleseed (дата обращения 16.05.2020)
14. Michael Oren and Shree K. Nayar. Generalization of lambert's reflectance model. // In Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '94, 239–246. 1994. ACM. URL: http://doi.acm.org/10.1145/192161.192213.
15. Brent Burley Extending the Disney BRDF to a BSDF with Integrated Subsurface Scattering // SIGGRAPH 2015. – 2015. – URL: https://blog.selfshadow.com/publications/s2015-shading-course/burley/s2015_pbs_disney_bsdf_notes.pdf (дата обращения 12.05.2020)
16. Diffuse BSDF Cycles Shader URL: https://docs.blender.org/manual/en/latest/render/shader_nodes/shader/diffuse.html (дата обращения 15.05.2020)
17. Emission Cycles Shader URL: https://docs.blender.org/manual/en/latest/render/shader_nodes/shader/emission.html (дата обращения 15.05.2020)
18. Cook R.L., Torrance K.E. A Reflectance Model for Computer Graphics // ACM Transanction on Graphics. – 1982. – Vol. 1.
19. Christophe Schlick An inexpensive BRDF model for Physikally-based Rendering // Eurographics 94, Computer Graphics Forum. – 1994. – Vol.13, No. 3. – 233-246 P.
20. А. В. Боресков Программирование компьютерной графики. Современный OpenGL. – М.: ДМК Пресс. – 2019. – 372 с.: ил.
21. Glossy BSDF Cycles Shader URL: https://docs.blender.org/manual/en/latest/render/shader_nodes/shader/glossy.html (дата обращения 15.05.2020)
22. Glass BSDF Cycles Shader URL: https://docs.blender.org/manual/en/latest/render/shader_nodes/shader/glass.html (дата обращения 15.05.2020)
23. Principled BSDF Cycles Shader URL: https://docs.blender.org/manual/en/latest/render/shader_nodes/shader/principled.html (дата обращения 15.05.2020)

24. Refraction BSDF Cycles Shader URL: https://docs.blender.org/manual/en/latest/render/shader_nodes/shader/refraction.html (дата обращения 15.05.2020)
25. Transparent BSDF Cycles Shader URL: https://docs.blender.org/manual/en/latest/render/shader_nodes/shader/transparent.html (дата обращения 15.05.2020)
26. LuxCoreRender Materials Glass URL: https://wiki.luxcorerender.org/LuxCoreRender_Materials_Glass#Reflection_Color (дата обращения 16.05.2020)
27. LuxCoreRender Materials Matte URL: https://wiki.luxcorerender.org/LuxCoreRender_Materials_Matte (дата обращения 16.05.2020)
28. LuxCoreRender Materials Matte Translucent URL: https://wiki.luxcorerender.org/LuxCoreRender_Materials_Matte_Translucent (дата обращения 16.05.2020)
29. LuxCoreRender Materials Mirror URL: https://wiki.luxcorerender.org/LuxCoreRender_Materials_Mirror (дата обращения 16.05.2020)
30. LuxCoreRender Materials Glossy URL: https://wiki.luxcorerender.org/LuxCoreRender_Materials_Glossy (дата обращения 16.05.2020)
31. LuxCoreRender Materials Glossy Translucent URL: https://wiki.luxcorerender.org/LuxCoreRender_Materials_Glossy_Translucent (дата обращения 16.05.2020)
32. LuxCoreRender Materials Metal URL: https://wiki.luxcorerender.org/LuxCoreRender_Materials_Metal (дата обращения 16.05.2020)
33. asDisneyMaterial appleseed Shader URL: https://appleseed.readthedocs.io/projects/appleseed-maya/en/latest/shaders/material/as_disney_material.html?highlight=Disney#asdisneymaterial (дата обращения 16.05.2020)
34. László Szirmay-Kalos Monte-Carlo Methods in Global Illumination // VDM Verlag Dr. Müller. – 2008. – 136 p. - ISBN-10: 9783836479196 URL: https://cg.iit.bme.hu/~szirmay/script.pdf (дата обращения 16.05.2020)
35. F. J. Medina-Aguayo, R. G. Everitt Revisiting the balance heuristic for estimating normalising constants // Cornell University. – 2019. – URL: https://arxiv.org/pdf/1908.06514.pdf

36. PhotonGI cache LuxCoreRender URL: https://wiki.luxcorerender.org/PhotonGI (дата обращения 17.05.2020)
37. Peak signal-to-noise ratio URL: https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio (дата обращения 17.05.2020)
38. skimage.metrics.peak_signal_noise_ratio(…) URL: https://github.com/scikit-image/scikit-image/blob/181788f5ae16535b560e3bec44033a5b3a52cdb6/skimage/metrics/simple_metrics.py#L108 (дата обращения 17.05.2020)
39. Metropolis, N., A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of state calculations by fast computing machines // Journal of Chemical Physics. – 1953. – Vol. 21 (6). – doi: 10.1063/1.1699114

Приложение А

1. Псевдокод алгоритма Path-tracing

```
Trace(ray){
    (object, vec_x) = FirstIntersect(ray)
    if !(intersection) // нет пересечения
        return L_sky

    color = L_e(vec_x, -ray.direction ) + DirectLightsource(vec_x, -ray.direction)
    prob = BRDFSampling(-ray.direction, normal, newray)
    if (prob == 0)
        return color

    color += Trace(newray) * w(newray.direction, normal, -ray.direction)/prob
    return color
}
```

1. Псевдокод алгоритма SPPM

```
void SPPMIntegrator::Render(const Scene &scene) {
    // Initialize pixelBounds and pixels array for SPPM
       Bounds2i pixelBounds = camera->film->croppedPixelBounds;
       int nPixels = pixelBounds.Area();
       std::unique_ptr<SPPMPixel[]> pixels(new SPPMPixel[nPixels]);
       for (int i = 0; i < nPixels; ++i)
           pixels[i].radius = initialSearchRadius;

    // Вычисление lightDistr для испускания фотонов пропорционально силе источника света
       std::unique_ptr<Distribution1D> lightDistr = ComputeLightPowerDistribution(scene);

    // Perform nIterations of SPPM integration
       HaltonSampler sampler(nIterations, pixelBounds);
       // Вычисление кол-ва плиток для использования SPPM camera pass
       for (int iter = 0; iter < nIterations; ++iter) {
           // генерация SPPM видимых точек --------------
              std::vector<MemoryArena> perThreadArenas(MaxThreadIndex());
              ParallelFor2D(
                  [&](Point2i tile) {
                      MemoryArena &arena = perThreadArenas[ThreadIndex];
                      // Follow camera paths for tile in image for SPPM
                  }, nTiles);
           // Создание сетки всех видимых точек
                  // Allocate grid for SPPM visible points
                  // Compute grid bounds for SPPM visible points
                  // Compute resolution of SPPM grid in each dimension
                  // Add visible points to SPPM grid

           // Отслеживание траектории фотонов и объединение вклада в эя
              std::vector<MemoryArena> photonShootArenas(MaxThreadIndex());
              ParallelFor(
                  [&](int photonIndex) {
                      MemoryArena &arena = photonShootArenas[ThreadIndex];
                      // Следование по пути фотона для photon_Index
                      arena.Reset();
                  }, photonsPerIteration, 8192);


           // Обновление значения пикселей из этого прохода фотонов
              for (int i = 0; i < nPixels; ++i) {
                  SPPMPixel &p = pixels[i];
                  if (p.M > 0) {
                      // обновление кол-ва фотонов, радиуса поиска ближайших фотонов
```

```
                        p.M = 0;
                        for (int j = 0; j < Spectrum::nSamples; ++j)
                            p.Phi[j] = (Float)0;
                    }
                    // сброс VisiblePoint в пикселе
                       p.vp.beta = 0.;
                       p.vp.bsdf = nullptr;

                }


            // Периодическое сохранение изображения SPPM в film
                    if (iter + 1 == nIterations || ((iter + 1) % writeFrequency) == 0) {
                    int x0 = pixelBounds.pMin.x;
                    int x1 = pixelBounds.pMax.x;
                    uint64_t Np = (uint64_t)(iter + 1) * (uint64_t)photonsPerIteration;
                    std::unique_ptr<Spectrum[]> image(new Spectrum[pixelBounds.Area()]);
                    int offset = 0;
                    for (int y = pixelBounds.pMin.y; y < pixelBounds.pMax.y; ++y) {
                        for (int x = x0; x < x1; ++x) {
                            // вычисление эя L для пикселя SPPM
                            image[offset++] = L;
                        }
                    }
                    camera->film->SetImage(image.get());
                    camera->film->WriteImage();
                }

        }

}
```

3. Код нормирования яркости с помощью модуля cv2

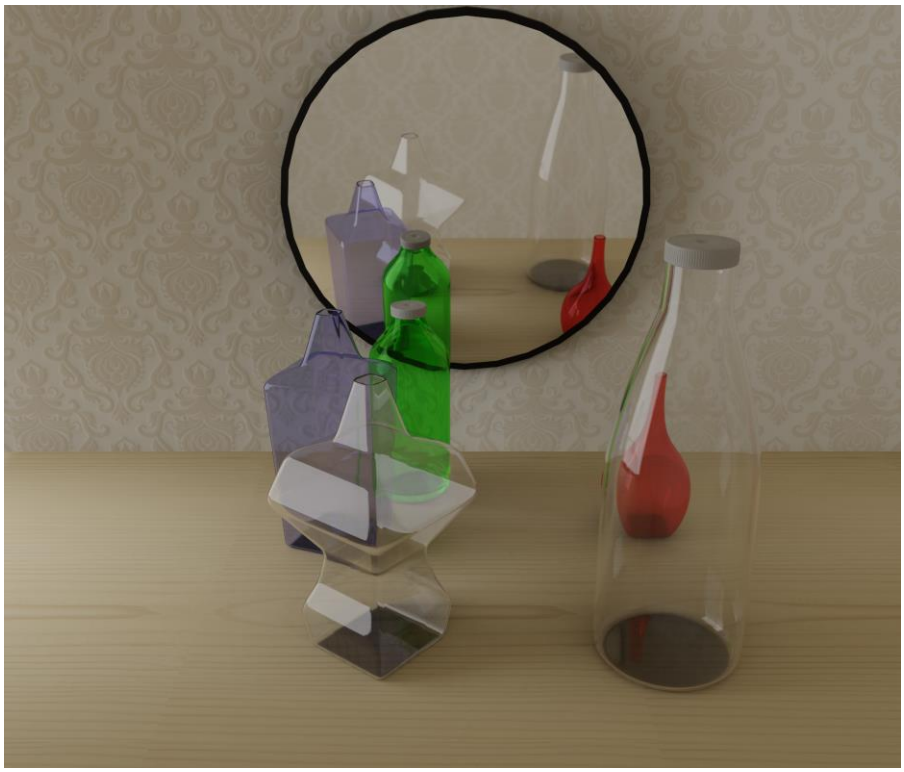
```
def hisEqulColor(img):
    ycrcb=cv2.cvtColor(img,cv2.COLOR_BGR2YCR_CB)
    channels=cv2.split(ycrcb)
    cv2.equalizeHist(channels[0],channels[0])
    cv2.merge(channels,ycrcb)
    return img
```

4. Получившиеся в результате экспериментов изображения доступны по ссылке: https://github.com/goginyanboris/VKR/tree/3d_experiments.

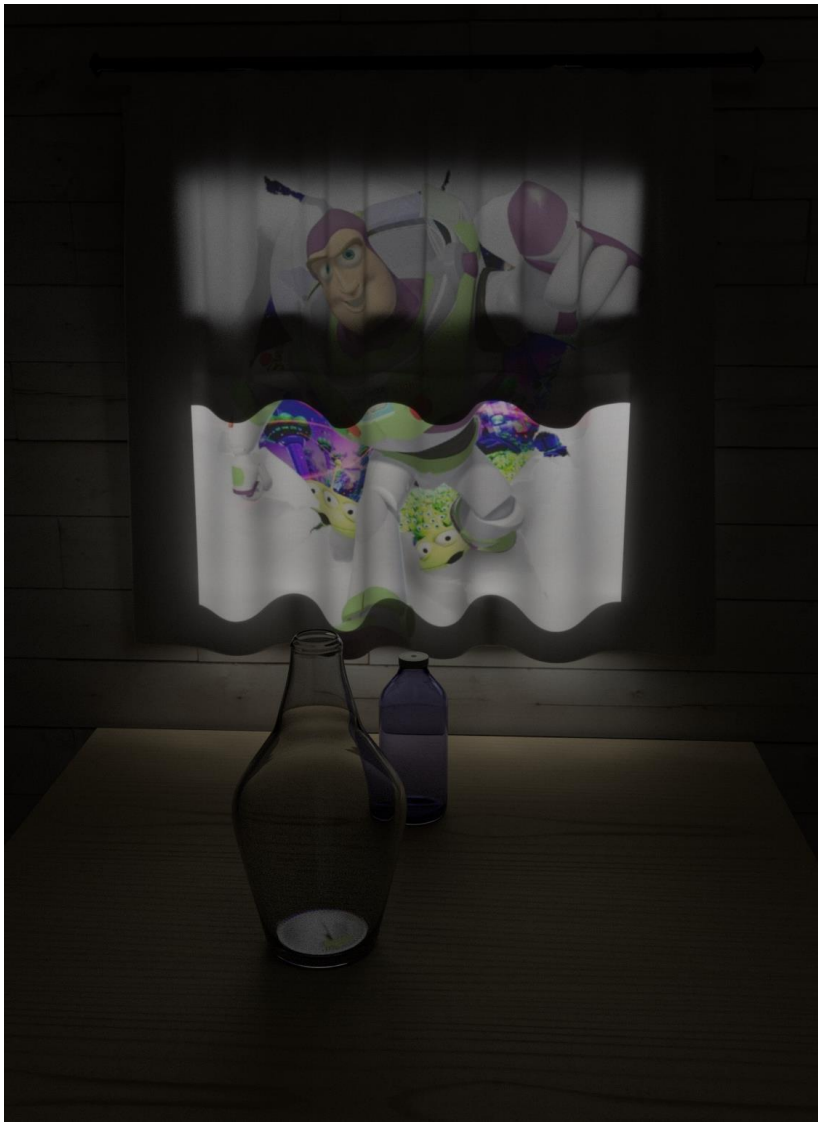
Приложение В

В1. Эталонное изображение 1 эксперимента

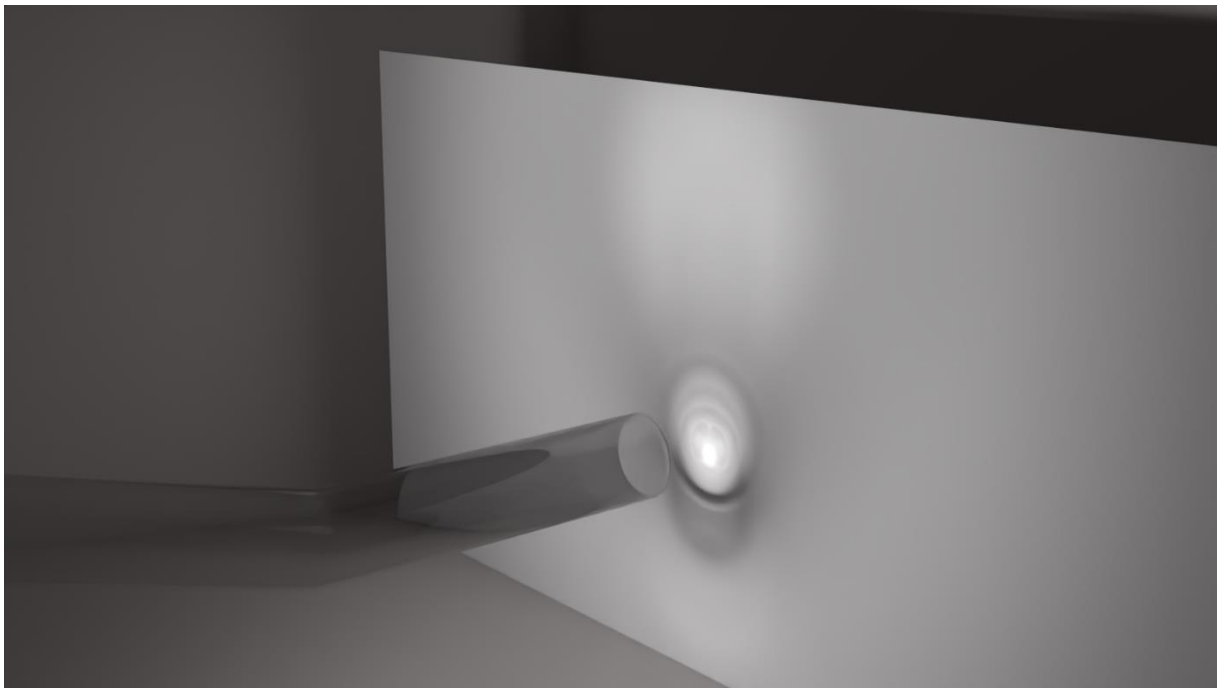
В2. Эталонное изображение 2 эксперимента



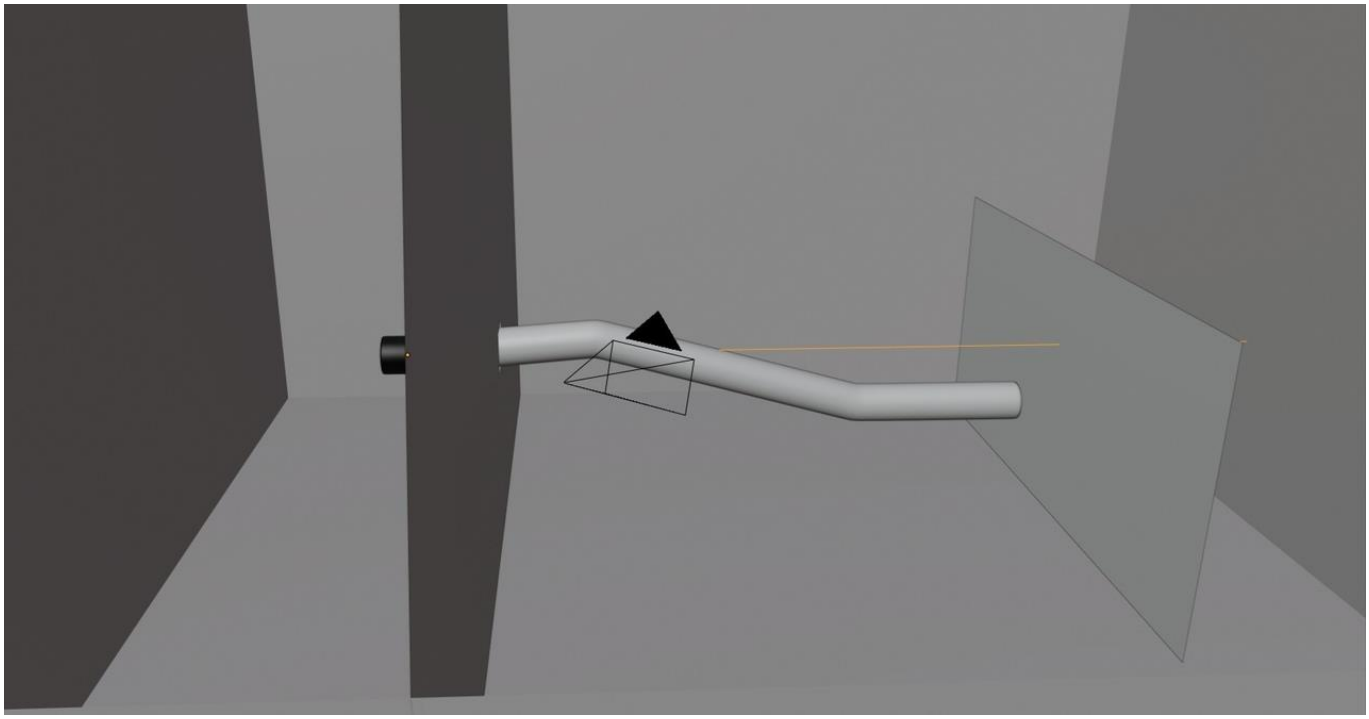
В3. Эталонное изображение 3 эксперимента



В4. Эталонное изображение 4 эксперимента



В5. 3D модель 4 эксперимента



В6. Эталонное изображение 5 эксперимента.